PCT/GB00/02587

11,1

PA 319534

THE UNITED STATES OF AMERICA

TO ALL TO WHOM THESE PRESENTS SHALL COME:

UNITED STATES DEPARTMENT OF COMMERCE

4

United States Patent and Trademark Office

GB00/02587

October 23, 2000

THIS IS TO CERTIFY THAT ANNEXED HERETO IS A TRUE COPY FROM THE RECORDS OF THE UNITED STATES PATENT AND TRADEMARK OFFICE OF THOSE PAPERS OF THE BELOW IDENTIFIED PATENT APPLICATION THAT MET THE REQUIREMENTS TO BE GRANTED A FILING DATE UNDER 35 USC 111.

APPLICATION NUMBER: *60/142,633*
FILING DATE: *July 06, 1999*

PRIORITY DOCUMENT

SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH RULE 17.1(a) OR (b)

By Authority of the
COMMISSIONER OF PATENTS AND TRADEMARKS

T. Lawrence

T. LAWRENCE
Certifying Officer

07/06/99 JC541 U.S. PTO

PTO/SB/16 MODIFIED BY AT&T CORP.

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

A/PRO

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53(c).

Docket Number	1999-0382

INVENTOR(S)

Name (line 1)	Stafford-Fraser, James Quentin
Address (line 1)	10 Marlborough Court
Address (line 2)	Cambridge
Address (line 3)	ENGLAND
Address (line 4)	CB3 9BQ
Address (line 5)	
Zip Code	

☐ *Additional Inventors are being named on the separately numbered sheets attached hereto*

07/06/99 JC541 U.S. PTO 60/142630

TITLE OF THE INVENTION (180 characters max)

Broadband Phone

CORRESPONDENCE ADDRESS

☐ ☐ *Customer Number or Bar Code Label* (Insert Customer No. or Attach bar code label here) *or* ☒ *Correspondence address below*

NAME	Samuel H. Dworetsky				
ADDRESS	AT&T CORP. P.O. Box 4110				
CITY	Middletown	STATE	New Jersey	ZIP CODE	07748-4801
COUNTRY	United States of America			FAX	732-368-6932

ENCLOSED APPLICATION PARTS *(check all that apply)*

☒ Specification *Number of Pages* 159 ☐ Small Entity Statement ☐ Other (specify)

☐ Drawing(s) *Number of Sheets* ☒ Return Receipt Postcard (MPEP 503)

METHOD OF PAYMENT OF FILING FEES FOR THIS PROVISIONAL APPLICATION FOR PATENT *(check one)*

The Commissioner is hereby authorized to charge filing fees or credit any overpayment to Deposit Account Number: 01-2745

Filing Fee Amount ($): $150.00

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

☒ No.

☐ Yes, the name of the U.S. Government agency and the Government contract number are :

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT REQUIRED

NAME	Samuel H. Dworetsky	Reg. #	27873
TELEPHONE	973-360-8120		
SIGNATURE		DATE	07/06/1999

"Express Mail" Mailing Label Number EL246515918US Date of Deposit 07/06/1999

I hereby certify that this provisional application is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Assistant Commissioner of Patents, Washington D.C., 20231

T. Phillips
(Printed Name of Person Mailing Paper)

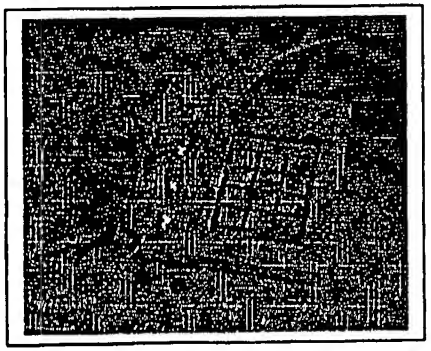
T Phillips
(Signature of Person Mailing Paper)

SEND TO: Box Provisional Application, Assistant Commissioner for Patents, Washington, DC 20231.

The Broadband Phone

Ever since the first phone call, smart people have worked hard to make communicating easy.

As we imagine the communications networks of the 21st century, AT&T Labs is thinking about how to help people stay in touch with other people, with information and with learning.

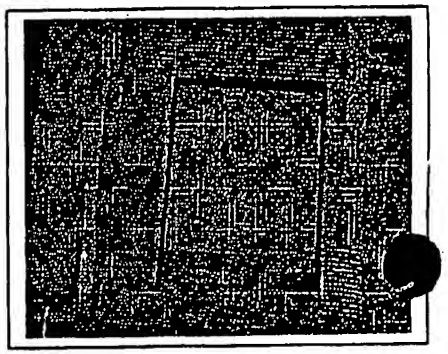


AT&T's Broadband Phone demonstrates that the power to control information need not be complicated or expensive. The most widely-accepted devices for managing tomorrow's broadband communications are likely to be familiar, intuitive, and easy to operate.

Today, about 50% of the households in the United States have a personal computer. Almost 100% have a telephone.

This working Broadband Phone is an end-to-end Internet Protocol (IP) device. It requires only a broadband connection and an ordinary DC power supply for the display. This example uses a 10 mb/s Ethernet connection to link to a server.

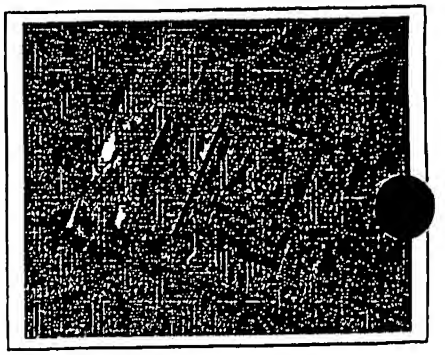
Instead of dial tone, the Broadband Phone uses tomorrow's IP tone. Pick up the handset or touch the display and the Broadband Phone is instantly on, connected to the network. Dial a number and connect to another Broadband Phone or IP device with the reliability and speed of the telephone network. Voice is sent over the IP network in digital, packetized form.



The display on the Broadband Phone is a simple, touch sensitive color screen that can be used with a stylus or finger.

Today's applications:

- Easily browse the web on your own, or share web pages.
- Order a pizza. Customize and place your order. See how long it will take to prepare.
- Scribble a note or a diagram that appears instantly to the person you are speaking with.
- Store your family photo album on the network. Search and retrieve images easily. Click on a thumbnail image to share a picture.
- Access your voice, email or fax messages. Scribble a reply and send it.



The Broadband Phone is inexpensive and simple to use because its applications are resident in the network, not in the phone. Technology developed at AT&T Laboratories in Cambridge, England uses IP to connect the simple display and touchscreen on the Phone to applications resident on an AT&T network server.

Because the applications are resident in the network, not the phone, they will be widely available and easy to update. Applications such as the ones shown today will not be limited to pairs of Broadband Phones: they will be accessed by any type of IP access device running AT&T Labs' open standard, "thin client" software.

You can download a free version of the AT&T Labs' software that makes the Broadband Phone possible. It's available at http://XXXXXXXXX.

Simple phone. Smart network.

Bandwidth Everywhere

Most of the world's communications runs on high-speed optical fiber. Information travels at the speed of light.

Today's local access networks bring this traffic to a grinding halt. For the most part, the twisted pair, copper connections that connect telephones are slow. They were designed to handle low-bandwidth voice calls.

AT&T will offer the first real competitive choice for local communications. Using two-way, high-speed cable connections, AT&T will bring new broadband services to people's homes. These new services will be based on Internet Protocols (IP), the design point for next century communications.

Widely available broadband connections will give rise to a new generation of IP-based access devices in the form of personal digital assistants, wireless telephones, wireless cameras and home controllers.

The number and variety of these devices will bring a new challenge: keeping personal directories, messages, documents and other data consistent and up to date.

AT&T's global broadband network will link all sorts of wired and wireless IP-based devices, helping people manage and control information.

The Broadband Phone represents the first of a new generation of simple-to-use, broadband information devices that many companies will be bringing to market in the years ahead.

TeleGUI

A new space of services
AT&T Confidential – 1 March 1999
This version 15 June 1999

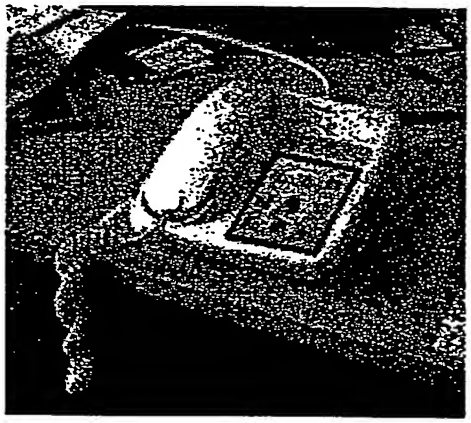
Quentin Stafford-Fraser

With the advent of high-bandwidth permanent connections to the home, a new paradigm for personal communications, and even for personal computing, becomes possible. In this new world, the network is everything. The commodity is no longer hardware or software, but services.

Some background

At AT&T Laboratories Cambridge, we have learned a great deal from the Virtual Network Computing project. The VNC system is based on the concept of ultra-thin clients which provide a framebuffer connected to a display, and have some devices, typically a keyboard and mouse, for input. These clients (or 'VNC viewers') connect to a remote VNC server, which uses very low-level graphics primitives to draw into the viewer's framebuffer, where the user can see the results. Typically, the server will draw a conventional workstation 'desktop' with which the user can interact and on which he can run standard applications. As an example, with the Java version of the VNC viewer and the Windows version of the VNC server, I can use a web browser to connect to a remote PC and run Windows applications.

The viewer is virtually stateless, with all important state stored at the server. This means that if you disconnect from the server and then reconnect, either from the same place or from elsewhere, the viewer simply requests a screen refresh, and you see the desktop exactly as you left it. Even the cursor will still be in the same place. VNC viewers and servers are available for a wide variety of platforms, from Cray mainframes to PalmPilots. The VNC system is freely available from our web site[1] and is in use by hundreds of thousands of people world-wide.



The VNC protocol is sufficiently simple that it could easily be embedded in consumer electronics devices. This paper proposes that a low-level graphics protocol such as VNC could be fed into the home or business and that this would provide a very powerful development of the telephone service as we know it.

The Communicator

For the purposes of this discussion we might imagine the 'Communicator' device pictured above: a telephone with its numeric keypad replaced by a touch screen which could, by default, display the old keypad and operate like a traditional

---

1 See http://www.uk.research.att.com/vnc. Detailed documentation on VNC's operation is also available from here.

telephone[2]. Many other form factors are also possible. We might consider wall-mounted Communicators with larger screens and hands-free operation, Communicators implemented as set-top boxes, Communicators which sit on a desk with a keyboard and mouse attached, or software versions which run on a PC. But whatever the form, we imagine a device which is at least as ubiquitous as the telephone, and in an ideal world, as ubiquitous as the light switch.

The Communicator's connection to the network might be ATM, IP or simple ISDN, and the VNC protocol, or something built on similar principles, would be carried alongside any voice traffic. The important point here is that this is not a phone with a PC in it, and it is not a phone with a web browser in it. It is a phone with a *display* in it. And every pixel of that display is driven over the network. We'll return to this point later, but first let's consider some simple examples of how such a device might be used:

- You telephone a business, and as an alternative to navigating through a list of voice prompts, you can select from an on-screen menu. While you are on hold, you might be shown information about the company's products, or about the person you will eventually be talking to. When you are connected, you might see a picture or the business card of the person at the other end.

- You don't need to remember arcane sequences of key presses to access the call-waiting, call-forwarding and similar facilities included with your telephone service, because you are prompted with a friendly on-screen menu.

- When you buy a new Communicator, you don't need to program it with all your speed-dial numbers. The service provider stores those for you, and displays them according to your preferences. You just plug it in. And if you use your AT&T calling card in a phone anywhere else, you get the same menus that you're used to at home.

- When you call a friend, you might, by default, get a shared sketch pad on which you could both make notes, draw maps, etc. while talking.

The telephone occupies a different place in the home from the TV or the PC, and the applications that people will want to use from a device that sits on their coffee table or bedside table may be very different from those used elsewhere. One might imagine a TV guide which is customised for your channels, which highlights your favourite programmes, and which shows low-frame-rate thumbnails of the current content of each video channel. Reminder services, directory information and yellow pages might all be given friendlier interfaces. And wouldn't it be easier to order pizza over the phone if you could look at the menu while you were doing it?

Why not browse?

There is an obvious comparison to be made with the World Wide Web. Many of these services can already be found there, and it could be argued that the sensible thing to put into the Communicator is a web browser, not a VNC viewer, because a browser generally requires less bandwidth and because the endpoint is arguably the thing that knows best how to render information effectively onto itself.

The bandwidth problem is a current one, more than a future one. We are anticipating home networks many many times faster than the modems which are currently used by most users for web surfing. We anticipate, from our VNC experience, that a bandwidth of around 100kb/s would make an ideal service, and many cable modems are now operating at

[2] Since the initial version of this paper, a prototype of such a device has been built at our laboratory.

nearly 300 times that rate. We have become very accustomed to saying "Processing power is cheap, so let's put a processor inside". I think we are entering a world where "Bandwidth is cheap, so let's use the network!" may be an equally valid assertion for many applications.

Any browser, in the end, has to base its rendering on the width, height and colour capabilities of the display that the user is sitting in front of. Whether it is directly connected to that display, or connected via a network is largely irrelevant. And all we are doing here is adding an extra level of indirection. It may be that when you buy the latest Communicator from ACME, Inc. you are also buying a subscription to their browser service, which is optimised for this particular device.

Even an application as simple as a web browser will always have compatibility and support issues, both of hardware and software, when you use it as the main communication device for millions of people. And this is not only when downloading plug-ins for exotic formats. How much memory does your Communicator have? That will affect how large a page it can download.

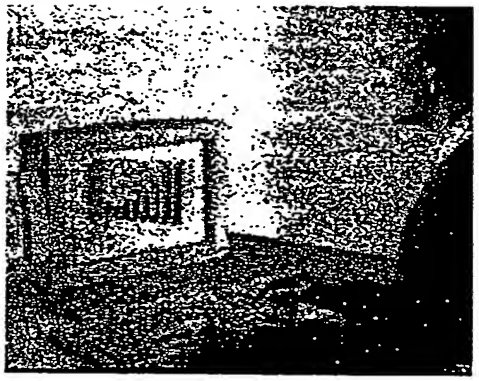
To put a browser into the Communicator would be a bit like putting a speech synthesis chip into a phone. Imagine if, when you connected to an information service, the information was sent to your phone as textual data and then converted to audio by the chip. What happens when different manufacturers' phones have different chips? Or when you connect to a foreign-language service which needs different phonemes? It's obviously much better to use the available bandwidth for direct end-to-end audio, because by going for the lowest common denominator you keep your future options open.

The same applies if you have enough bandwidth for end-to-end graphics. Imagine if you had built a phone based around a web browser of four years ago. It almost certainly would not have supported Java, or the 'frame' facility of HTML, which is now used by a large proportion of web sites. But if all you have in the home are the pixels, you can still provide a web browsing service, but you can update it at will without the user needing to be involved. I don't like to imagine a world where my father might have the same problems updating his phone that he has updating his PC!

The dumber the home device, and the more we can centralise the sophisticated stuff, the easier the maintenance will be and the less we will suffer from the need to update the endpoints.

The ability to enhance telephone services and to provide web access from the coffee table introduces a large number of new possibilities, but this is only the tip of the iceberg.

The office Communicator



We have often described the VNC protocol as being marginally higher-level than a VGA lead. The viewer has no knowledge of the semantic structure of the data being sent to it; it just displays pixels. Mouse and keyboard events are sent back simply as coordinates and button-presses. In many ways, VNC simply allows you to use the internet as an arbitrarily long keyboard & monitor extension lead.

Now imagine a business model based on this concept. At present, in many specialised business areas, computers are bought as part of a package. The supplier installs them in your premises, with the particular software package installed,

configured and ready to run. When the software or hardware needs upgrading, they come back to your office and do it for you. Very convenient.

But how much more convenient if they never had to visit you to set it up in the first place! If you could just connect the monitor, keyboard and printer on *your* desk to the computer while it was still in *their* office. If they handled all your backups for you, all your software upgrades, and did it overnight or at the weekend when you didn't need it. If they had plenty of backup machines that could be dropped into place in a matter of minutes if yours died.[3] With a system like VNC over a network as reliable as the phone network, this type of business would be possible.

This one service provider might only specialise in one type of software, but that wouldn't matter. You would 'dial up' WordPerfect Inc. when you wanted to do some word-processing, Qualcomm when you wanted to read your email, and Netscape when you wanted to browse the web. Each of these companies would provide a service, very like a premium-rate phone call, and they would drive your screen, or part of it, while you were connected. Some companies might provide free word-processing facilities in exchange for a little advertising space on your screen, in the same way that free email services currently do on the web. If you want to try out a piece of software to see whether it's suitable for your needs, you don't need to go through all the palaver of installing it on your machine and uninstalling it afterwards. You just dial it up on your display.

This is the ultimate in outsourcing. No software is downloaded to the user, there are no problems with the user's machine being insufficiently powerful or being incompatible in some way, and there are no opportunities for software piracy or viruses. This is, in fact a whole new class of services, and potentially a whole new model for the computing business.

[3] Of course, in a real implementation, you probably would not use a single individual machine per customer, but it's a good model!

Interesting questions:

Here are some interesting starting points for discussion. None of these would need to be decided immediately. One big advantage of dumb endpoints is that many decisions can be delayed and implemented later – they don't have to be in the device from the outset.

1. Do all phones in one house display the same thing, or are they effectively separate lines?

2. Do mobile/untethered phones include a display? If it's in the handset, you can't see it while talking. If it's on the wall, it might want to display the session of the nearest handset, and so would need to know which that is. Maybe the display should also be mobile.

3. Do the display devices in the Communicator also operate other household electronics? Can you use them to program your burglar alarm, your video and your microwave? These devices would not then need expensive displays of their own, but might need to be connected to a home 'exchange' (ATM switch?). You could even, then, have another class of services where remote 'experts' dial in to your home devices and configure them for you.

4. Lastly, do we expect the Communicator to display video? This is certainly possible. And would there be an option of high-quality audio so that it could become an extension of the hi-fi system and a replacement for the radio?

These questions are all related to the user's experience. There are, of course, plenty of issues which the service providers would also have to tackle, such as security, confidentiality, and reliability. Allocation

of processing power to drive the devices might also be an issue, but if the network is in place and the network is reliable, then many issues can be decoupled from the provision of the network and opened up to the market. As a user, if my web browser is too slow I simply dial up another one. As a provider, if too many users want to connect to my service at once then I just buy in more processing power from another company and forward my connections there in the same way that electricity providers cope with regional surges in demand.

Summary

Most businesses depend heavily on the telephone network and are, in effect, unable to operate without it. The challenge is to make a high-bandwidth IP network which is as reliable. If this can be achieved, then suddenly your external connectivity becomes more dependable than your internal connectivity. Most corporate computer networks suffer from substantial downtimes when compared to the public telephone network. These days, in fact, it is probably not actually the 'network' which has the problem. Corporate downtime is more likely to come from problems with the servers (eg. hardware and software upgrades) than with the connections to them. The question is, can a public network service do better?

The company which can first build an affordable wide-area network which is reliable and fast enough to be, in effect, the connection between a user's monitor and his CPU, has the potential dramatically to alter the business and computing world.

This does not necessarily mean that the connection has to be as reliable as a VGA lead, but it does mean that the combination of local devices, remote services, and the link between them must be demonstrably as reliable and cost-effective as the PC connected to a corporate network. This, I believe is achievable, if the customer equipment is kept simple, and if there is free and open competition in the service-provision market.

And at the very least we ought to be able to improve the user interface of the telephone system!

VIRTUAL NETWORK COMPUTING

TRISTAN RICHARDSON, QUENTIN STAFFORD-FRASER, KENNETH R. WOOD, AND ANDY HOPPER*
The Olivetti & Oracle Research Laboratory

VNC is an ultra-thin client system based on a simple display protocol that is platform-independent. It achieves mobile computing without requiring the user to carry any hardware.

The so-called network computer (NC) aims to give users access to centralized resources from simple, inexpensive devices. These devices act as clients to more powerful server machines that are connected to the network and provide applications, data, and storage for a user's preferences and personal customizations. We have taken this idea a stage further. In the virtual network computing (VNC) system, server machines supply not only applications and data but also an entire desktop environment that can be accessed from any Internet-connected machine using a simple *software* NC. Whenever and wherever a VNC desktop is accessed, its state and configuration (right down to the position of the cursor) are exactly the same as when it was last accessed.

In contrast to many recent Internet applications, which have focused on giving users access to resources located anywhere in the world from their home computing environments, VNC provides access to home computing environments from anywhere in the world. Members of the Olivetti & Oracle Research Laboratory (ORL) use VNC to access their personal Unix and PC desktops from any office in our Cambridge building and from around the world on whatever computing infrastructure happens to be available—including, for example, public Web-browsing terminals in airports. VNC thus provides mobile computing without requiring the user to carry any device whatsoever. In addition, VNC allows a single desktop to be accessed from several places simultaneously, thus supporting appli-

*Andy Hopper is also affiliated with Cambridge University Engineering Department.

1089-7801/98/$10.00 ©1998 IEEE

THE VNC PROTOCOL

The technology underlying the VNC system is a simple protocol for remote access to graphical user interfaces. It works at the framebuffer level and therefore applies to all operating systems, windowing systems, and applications—indeed to any device with some form of communications link. The protocol will operate over any reliable transport such as TCP/IP.

The endpoint with which the user interacts (that is, the display and/or input devices) is called the *VNC client* or *viewer*. The endpoint where changes to the framebuffer originate (that is, the windowing system and applications) is known as the *VNC server* (see Figure 1).

VNC is truly a "thin-client" system. Its design makes very few requirements of the client, and therefore simplifies the task of creating clients to run on a wide range of hardware.

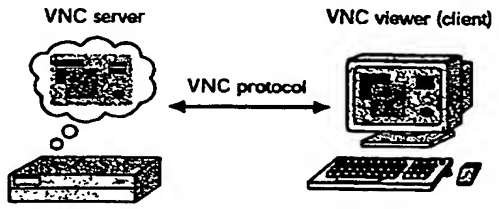


Figure 1. VNC architecture.

A Single Graphics Primitive

The display side of the protocol is based on a single graphics primitive:

> Put a rectangle of pixel data at a given *x*, *y* position.

At first glance this might seem an inefficient way to draw some user interface components. However, allowing various encoding schemes for the pixel data gives a large degree of flexibility in trading off parameters such as network bandwidth, client drawing speed, and server processing speed.

The lowest common denominator is the so-called *raw encoding*, where the pixel data for a rectangle is simply sent in left-to-right scanline order. All VNC clients and servers must support this encoding. However, the encodings actually used on a given connection can be negotiated according to the capabilities of the server and client and the connection between them.

For example, *copy-rectangle encoding* is very simple and efficient, and can be used when the client already has the same pixel data elsewhere in its framebuffer. The encoding on the wire is simply an *x*, *y* coordinate. This gives a position in the framebuffer from which the client can copy the rectangle of pixel data. This encoding is typically used when the user moves a window across the screen or scrolls a window's contents.

Most clients will support copy-rectangle encoding, since it is generally easy to implement, saves bandwidth, and is likely to be faster than sending raw data again. However, in a case where a client cannot easily read back from its framebuffer, the client could specify that it should *not* be sent data encoded this way.

A typical workstation desktop has large areas of solid color and text. One of our most effective encodings takes advantage of this phenomenon by describing rectangles consisting of one majority (background) color and "sub-rectangles" of different colors. There are numerous other possible schemes. We could use a JPEG encoding for efficient transmission of still images or an MPEG encoding for moving images. A pixel-data caching scheme could efficiently encode multiple occurrences of the same text character by referring to the first occurrence.

Adaptive Update

A set of rectangles of pixel data makes a *framebuffer update* (or simply, *update*). An update represents a change from one valid framebuffer state to another. In this sense, an update is similar to a frame of video. It differs, however, in that it usually affects only a small area of the framebuffer. Each rectangle may be encoded using a different scheme. The server can therefore choose the encoding most appropriate for the particular screen content being transmitted and the available network bandwidth.

The update protocol is demand-driven by the client. That is, an update is only sent by the server in response to an explicit request from the client. All screen changes since the client's last request are coalesced into a single update. This gives the protocol an adaptive quality: the slower the client and the network, the lower the rate of updates. On a fast network, for example, as the user drags a window across the screen it will move smoothly, being drawn at all the intermediate positions. On a slower link—for example, over a modem—the client will request updates less frequently, and the window will appear at fewer of these positions. This means that the display will reach its final state as quickly as the network bandwidth will allow, thus maximizing the speed of interaction.

Input

The input side of the VNC protocol is based on a standard workstation model of a keyboard and multibutton pointing device. The client sends input events to the server whenever the user presses a key or pointer button, or moves the pointing device. Input events can also be synthesized from other nonstandard I/O devices. On the Videotile, for example, a pen-based handwriting recognition engine generates keyboard events.

Connection Setup and Shutdown

To establish a client-server connection, the server first requests authentication from the client, using a challenge-response

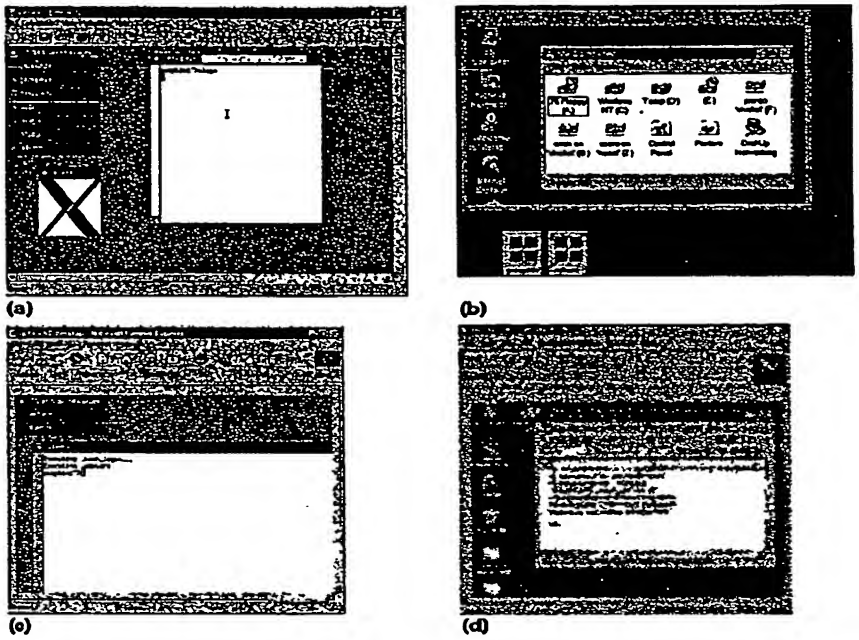


Figure 2. A variety of desktops being accessed from different viewers: (a) a Unix desktop from a Windows viewer, (b) a Windows 95 desktop from an X viewer, (c) a Unix desktop from a Java applet within Internet Explorer, and (d) a Windows desktop using Netscape on Unix.

scheme; the client typically requires the user to enter a password at this point. The server and client then exchange messages to negotiate desktop size, pixel format, and encoding schemes. The client requests an update for the entire screen, and the session begins. Because of the stateless nature of the client, either side can close the connection at any time without adverse consequences.

VNC Viewers

In day-to-day use, we prefer the more descriptive term *viewer* to the rather overloaded word *client*. Writing a VNC viewer is a simple task, as indeed it should be for any thin-client system. It requires only a reliable transport (usually TCP/IP), and a way of displaying pixels (either writing directly to the framebuffer or going through a windowing system).

We have written viewers for all the networked display devices available at ORL. These include the Videotile (the original VNC viewer), an X-based viewer (which runs on Solaris, Linux, and Digital Unix workstations), a Win32 viewer that runs on Windows NT and 95, and a Java applet that runs on any Java-capable browser (including Sun's JavaStation). Members of our lab use these viewers on a daily basis to access their personal computing environments.

The images in Figure 2 show a variety of X and Windows desktops being accessed from both Java and native X and Windows viewers.

VNC Servers

Writing a VNC server is slightly harder than writing a viewer. Because the protocol is designed to make the client as simple as possible, it is usually up to the server to perform any necessary translations (for example, the server must provide pixel data in the format the client wants). We have written servers for our two main platforms, X (that is, Unix) and Windows NT/95.

The X-based server was the first one we developed. A single Unix machine can run a number of VNC servers for different users, each representing a distinct VNC desktop. Each desktop is like a virtual X display, with a root window on which several X applications can appear.

The Windows VNC server was a little more difficult to create. Windows has fewer places to insert hooks into the system to monitor display updates, and the model of multiuser operation is less clearly defined. Our current server simply mirrors the real display to a remote client, which means that only a single VNC desktop is available from any one PC.

The X-based server, the X viewer, the Win32 server, and Win32 viewer can all fit on a single floppy disk.

We have also created "thin" servers which produce displays other than desktops, using a simple toolkit. A "VNC CD player," for example, generates a CD player user interface using VNC directly without any reference to a windowing system or framebuffer (see figure 3 on the following page). Such servers can run on very simple hardware, and can be accessed from any of the standard VNC viewers.

ANY USER INTERFACE, ANYWHERE

At ORL, we have used VNC to add mobility to workstation GUIs, where the concept of at least some form of remote interaction is not new. But the protocol's simplicity could allow it to be used on a much wider range of hardware. Consumer electronics devices, such as CD players, usually have a highly specialized user interface and typically employ customized phys-

REFERENCES

1. T. Richardson et al., "Teleporting in an X Window System Environment," *IEEE Personal Comm.*, No. 3, 1994, pp. 6-12. Also available as ORL Technical Report 94.4, ORL, Cambridge CB2 1QA, England.
2. T. Richardson, "Teleporting—Mobile X Sessions," *Proc. 9th Ann. X Technical Conf.*, Jan. 1995. Also in *The X Resource*, Issue 13, O'Reilly & Associates, Jan. 1995. Also available as ORL Technical Report 95.5, ORL, Cambridge CB2 1QA, England.
3. Open Group, "X11R6.3 (Broadway) Overview," http://www.opengroup.org/tech/desktop/x/broadway.html#bx (current September 1997).
4. K.R Wood et al., "Global Teleporting with Java: Toward Ubiquitous Personalized Computing," *Computer*, Vol. 30, No. 2., Feb. 1997, pp. 53-59. Also available as ORL Technical Report 96.2, ORL, Cambridge CB2 1QA, England.

Tristan Richardson is a research scientist at ORL, and his research interests include mobile and network computing, windowing systems, and multimedia. He holds an MA in computer science and an MPhil in computer speech and language processing from the University of Cambridge.

Quentin Stafford-Fraser is a research scientist at ORL, and his chief research interests are personalized mobile computing, in-car information systems, and novel user interfaces. Before joining ORL he worked at Rank Xerox EuroPARC (now XRCE) on video-augmented environments. He holds an MA and a PhD from the University of Cambridge.

Kenneth R. Wood is a research scientist at ORL. His interests include mobile computing, multimedia, concurrency theory, and applied formal methods. Before joining ORL, he worked as a member of the scientific staff at Nortel Technology and taught at Oxford University. He received the AB degree in applied mathematics from Harvard University and the MSc and DPhil degrees in computation from Oxford University.

Andy Hopper is director of the Olivetti & Oracle Research Laboratory (ORL) in Cambridge, a director of Advanced Telecommunications Modules Limited, and chair of Telemedia Systems Ltd. He is also a professor of communications engineering at the University of Cambridge and a fellow of Corpus Christi College. His research interests include networking, multimedia, and mobile systems. He received the BSc degree from the University of Wales in 1974 and the PhD degree from the University of Cambridge in 1978. He is a fellow of the IEE and the Royal Academy of Engineering.

Readers may contact the authors at ORL, 24a Trumpington Street, Cambridge CB2 1QA, UK; email: vnc@orl.co.uk.

Virtual Network Computing

AT&T

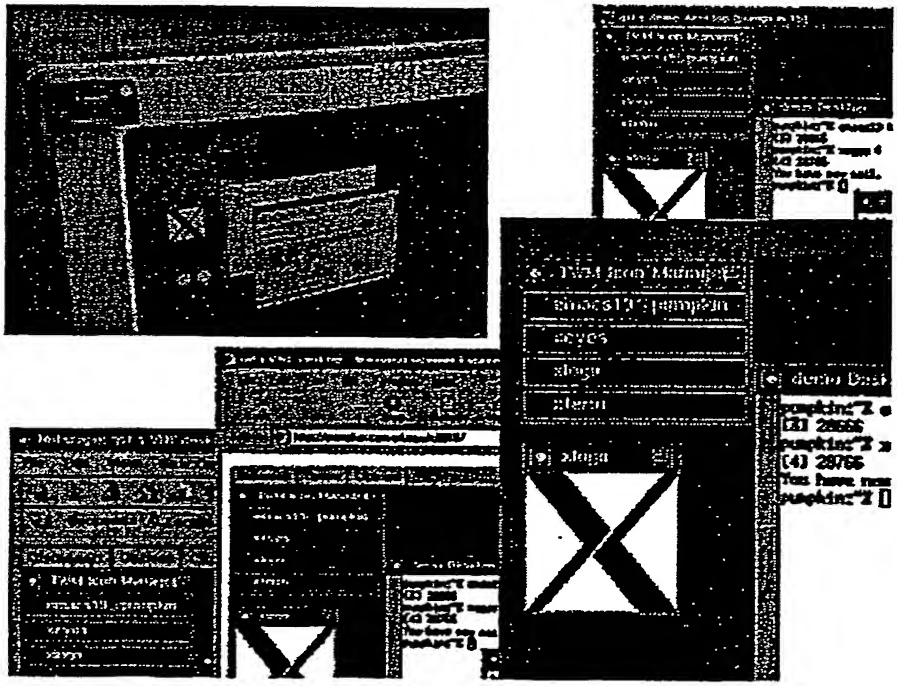
Home
Screenshots
Why is it free?
Getting started
Documentation
FAQ
Download
Keep in touch
Contributed
Want to help?
VNC people
Search
About it...

Some recent changes:

Macintosh software updated to Beta 2 - new documentation soon - 11/6/99
Experimental Amiga server - see the contribs pages- 10/6/99
Geos, Netwinder & NetBSD info added to the contribs pages- 27/4/99
VNC under new ownership! - 4/2/99
Note: The FAQ and some other bits of the documentation are constantly being updated. We only record major changes here.

What is VNC? - A practical introduction

VNC stands for Virtual Network Computing. It is, in essence, a remote display system which allows you to view a computing 'desktop' environment not only on the machine where it is running, but from anywhere on the Internet and from a wide variety of machine architectures.



The VNC system allows you to access the same desktop from a wide variety of platforms.

Many of us, for example, use a VNC viewer running on a PC on our desks to display our Unix environments which are running on a large server in the machine room downstairs.

What makes it different from other systems?

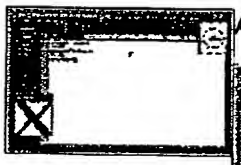
Virtual Network Computing

AT&T

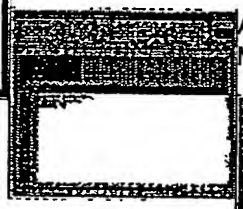
Home
Screenshots
Why is it free?
Getting started
Documentation
FAQ
Download
Keep in touch
Contributed
Want to help?
VNC people
Search
About Us

VNC screenshots

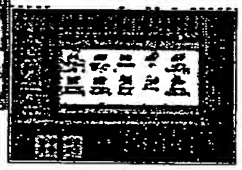
These shots show very simple desktops being accessed from a number of different platforms. Click on the thumbnails for larger images.



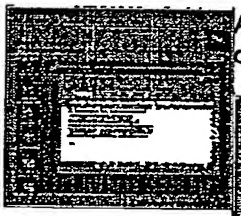
An X desktop being viewed from a native PC viewer.



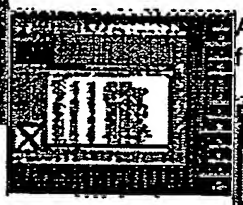
An X desktop being viewed from Microsoft Internet Explorer on a PC.



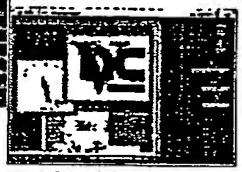
A Windows desktop being viewed from a native X viewer.



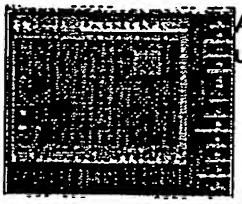
A Windows desktop being used from within Netscape on a Unix machine.



A Unix desktop being accessed from a Macintosh using Java.



A Unix desktop being accessed from a native Macintosh viewer.



A Windows desktop being accessed from a Macintosh using Java.

Virtual Network Computing

AT&T

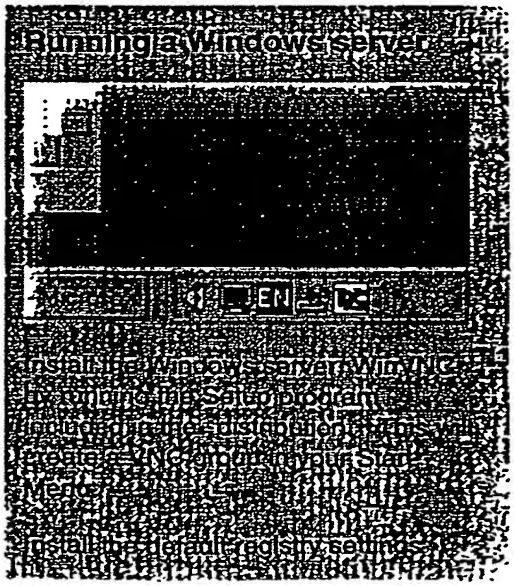
Getting Started with VNC

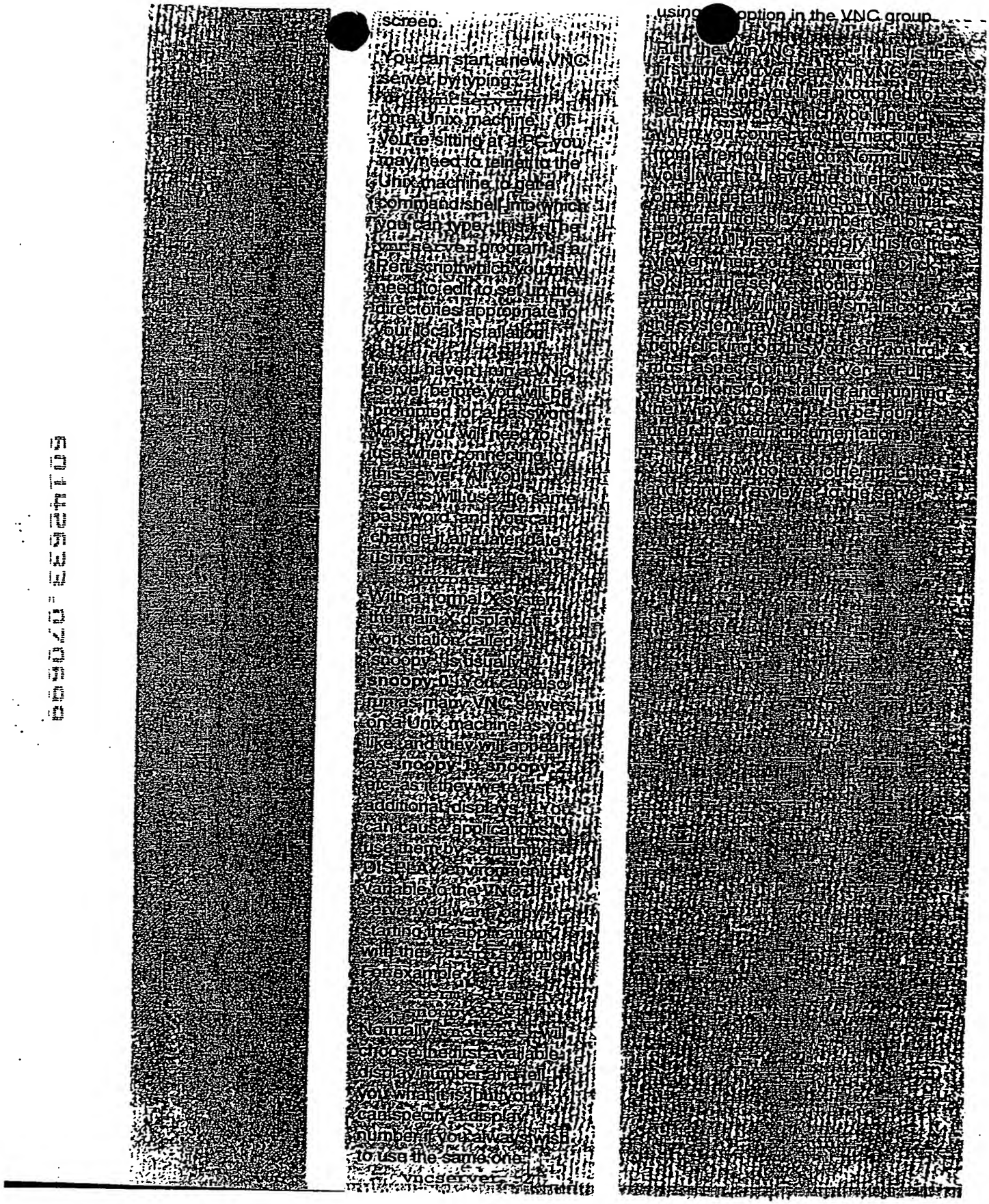
VNC consists of two types of component. A server, which generates a display, and a viewer, which actually draws the display on your screen. There are two important features of VNC:

- The server and the viewer may be on different machines and on different architectures. We expect the most common use to be the display of a Unix X desktop on a PC, for example. The protocol which connects the server and viewer is simple, open, and platform-independent.
- No state is stored at the viewer. Breaking the viewer's connection to the server and then reconnecting will not result in any loss of data. Because the connection can be remade from somewhere else, you have easy mobility.

So, to get started with VNC you need to run a server, and then connect to it with a viewer. Get the packages for the platforms you use from the download page, if you haven't already, and install them. The current VNC software requires a TCP/IP connection between the server and the viewer, though there is no reason why the software couldn't be modified to use, for example, RS232 or Firewire. We have internal versions that use other network transport layers. But for now you'll need to know the name or the IP address of the server machine.

Most people will be running either a Unix server or a Windows server, though similar principles will apply to other platforms.



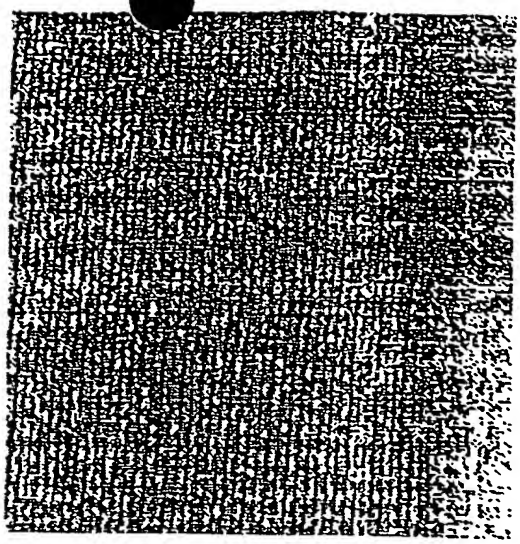


arting a server. To see anything you need to connect a viewer to the server. See below. The server will generate a log in your .vnc directory if you have problems at this stage, see the full documentation and the FAQ.

Killing a Unix server

You can kill a Unix VNC server using, for example:

```
vncserver -kill :2
```

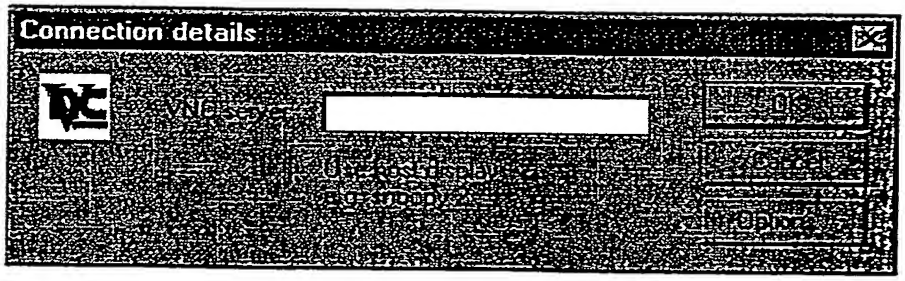


Running a viewer

When you run the viewer, you need to specify the name of the server and the number of the desktop. If, for example, you have started a server as display 2 on a machine called 'snoopy', you can start a viewer for it by typing:

```
vncviewer snoopy:2
```

With the Windows viewer, you can run it from the command line, but you will more typically run it from the VNC group on the Start Menu. In this case, you will be prompted for the host name and display number:

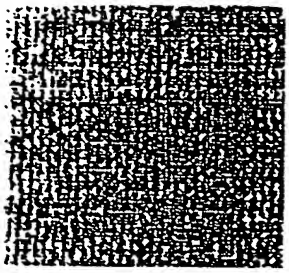


Enter it and click OK, and you will be prompted for your password, after which you should see the remote display. (If you are connecting to a Windows or Mac server, the display number will be 0, unless you have explicitly changed it).

If the machine running the server does not have a proper DNS entry, you probably won't be able to use the name and will have to replace `snoopy:2` with something like `192.168.1.2:2`. You can get round this on most platforms by creating a 'hosts' file which maps names onto IP addresses. Consult your local guru for help with this.

Using a web browser as a viewer

The VNC servers also contain a small web server. If you connect to this with a web browser, you can download the Java version of the viewer, and use this to view the server. You can then see your desktop from any



display 2 on machine 'snoopy', you wou int your web browser at:
`http://snoopy:5802/`
The applet will prompt you for your password, and should then display the desktop.

That's it! For more details see the documentation . The answers to lots of common questions can be found in the FAQ .

Virtual Network Computing

AT&T

VNC documentation

Note: A detailed paper about VNC can be found as:
Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood & Andy Hopper, "Virtual Network Computing", IEEE Internet Computing, Vol.2 No.1, Jan/Feb 1998 pp33-38.
You can download it in Acrobat format here (760k).

The following documentation assumes a basic familiarity with the terms and components used in VNC. See 'What is VNC? ' and 'Getting Started' for introductory information, and the Frequently Asked Questions for common queries.

For a quick overview, you can download versions of the VNC Video .

See also 'What's new in the VNC packages? '.

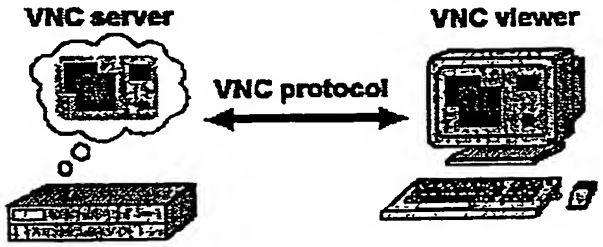
Technical documentation - Table of Contents

Home
Screenshots
Why is it free?
Getting started
Documentation
FAQ
Download
Keep in touch
Contributed
Want to help?
VNC people
Search
About us

VNC - How it works

The VNC Protocol

The VNC protocol is a simple protocol for remote access to graphical user interfaces. It is based on the concept of a remote framebuffer or RFB. In the past we have tended to refer to the VNC protocol as the RFB protocol, so you may have seen this term in other publications. The protocol simply allows a server to update the framebuffer displayed on a viewer. Because it works at the framebuffer level it is potentially applicable to all operating systems, windowing systems and applications. This includes X/Unix, Windows 3.1/95/NT and Macintosh, but might also include PDAs, and indeed any device with some form of communications link. The protocol will operate over any reliable transport such as TCP/IP.



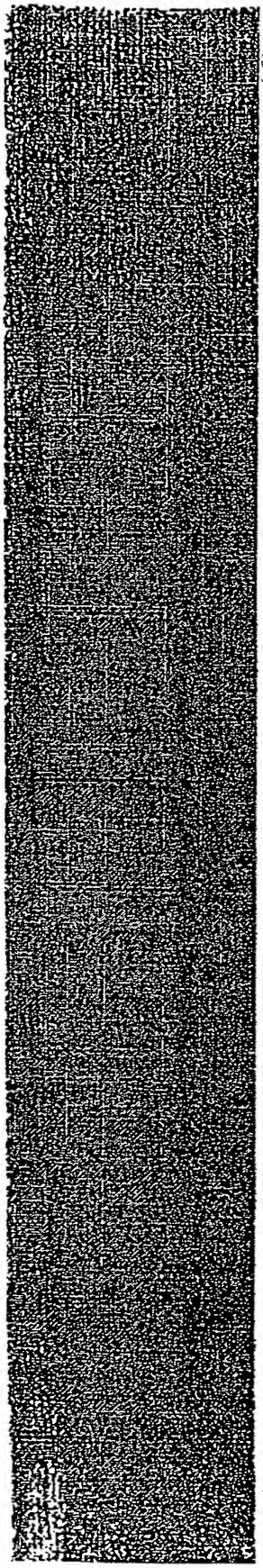
This is truly a "thin-client" protocol: it has been designed to make very few requirements of the viewer. In this way, clients can run on the widest range of hardware, and the task of implementing a client is made as simple as possible.

Rectangular updates

The display side of the protocol is based around a single graphics primitive: "put a rectangle of pixel data at a given x,y position". This might seem an inefficient way of drawing arbitrary user interface components. But because we have a variety of different encoding schemes for the pixel data, we can select the appropriate scheme for each rectangle we send, and make the most of network bandwidth, client drawing speed and server processing speed.

The lowest common denominator is the so-called raw encoding, where the rectangle is simply pixel data sent in left-to-right scanline order. All clients and servers must support this encoding. However, the encodings actually used on any given VNC connection can be negotiated according to the abilities of the server, the client, and the connection between the two.

The copy rectangle encoding, for example, is very simple and efficient



elsewhere in its framebuffer. The server simply sends an (x,y) coordinate giving the position from which the client can copy the rectangle of pixel data. This means that operations such as dragging or scrolling a window, which involve substantial changes to the screen, may only require a few bytes. Most clients will support this encoding, since it is generally simple to implement and saves bandwidth.

A typical workstation desktop has large areas of solid colour and of text. Some of our most effective encodings take advantage of this by efficiently describing rectangles consisting of one majority (background) colour and 'sub-rectangles' of different colours. There are numerous other possible schemes. We might use a JPEG encoding for still images or MPEG for efficient transmission of moving images. An encoding which uses some kind of caching of pixel data would be good for rendering text, where the same character is drawn in the same font multiple times. Subsequent occurrences of the same character would be encoded simply by reference to the first occurrence.

Adaptive update protocol

A sequence of these rectangles makes a framebuffer update (or simply update). An update represents a change from one valid framebuffer state to another, so in some ways is similar to a frame of video, but it is usually only a small area of the framebuffer that will be affected by a given update. Each rectangle may be encoded using a different scheme. The server can therefore choose the best encoding for the particular screen content being transmitted and the network bandwidth available.

The update protocol is demand-driven by the client. That is, an update is only sent by the server in response to an explicit request from the client. This gives the protocol an adaptive quality. The slower the client and the network are, the lower the rate of updates becomes. Each update incorporates all the changes to the 'screen' since the last client request. With a slow client and/or network, transient states of the framebuffer are ignored, resulting in reduced network traffic and less drawing for the client. This also improves the apparent response speed.

Input protocol

The input side of the protocol is based on a standard workstation model of a keyboard and multi-button pointing device. Input events are sent to the server by the client whenever the user presses a key or pointer button, or whenever the pointing device is moved. These input events can also be synthesised from other non-standard I/O devices. On our Videotile , for example, a pen-based handwriting recognition engine generates keyboard events.

Connection Setup and Shutdown

When the connection between a client and a server is first established, the server begins by requesting authentication from the client using a challenge-response scheme, which typically results in the user being prompted for a password at the client end. The server and client then exchange messages to negotiate desktop size, pixel format, and the encoding schemes to be used. The client then requests an update for

the entire screen, and the session begins. Because of the stateless nature of the client, either side can close the connection at any time without adverse consequences.

VNC Clients

Writing an VNC viewer is a simple task, as it should be for any thin-client system. It requires only a reliable transport (usually TCP/IP), and a way of displaying pixels (either directly writing to the framebuffer, or going through a windowing system).

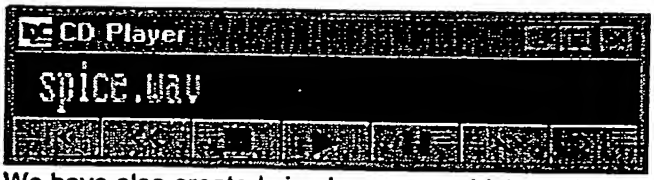
We have clients for all the networked display devices we have available at our lab. This includes the Videotile (the original RFB client), an X-based client (which runs on Solaris, Linux and Digital Unix workstations), a Win32 client which runs on Windows NT and 95, a Macintosh client, and a Java client which runs on any Java-capable browser (including Sun's JavaStation). Members of our lab use these clients on a daily basis to access their personal computing environments.

VNC Servers

Writing an VNC server is slightly harder than writing a client for a number of reasons. The protocol is designed to make the client as simple as possible, so it is usually up to the server to perform any necessary translations. For example, the server must provide pixel data in the format the client wants. We have servers for our two main platforms, X (i.e. Unix) and Windows NT/95.

A Unix machine can run a number of Xvnc servers for different users, each of which represents a distinct VNC desktop. Each VNC desktop is like a virtual X display, with a root window on which several X applications can be displayed.

The Windows server (WinVNC) is a little more difficult to create, because there are fewer places to insert hooks into the system to monitor display updates, and a less clearly-defined model of multiuser operation. Our current server simply mirrors the real display to a remote client, which means that the server is not 'multiuser'. It does, however, provide the primary user of a PC with remote access to their desktop.



We have also created simple servers which produce displays other than desktops, using a simple toolkit. A "VNC CD player", for example, generates a CD player user interface using the protocol directly without any reference to a windows system or framebuffer. Such servers can run on very simple hardware, and can be accessed from any of the standard viewers.

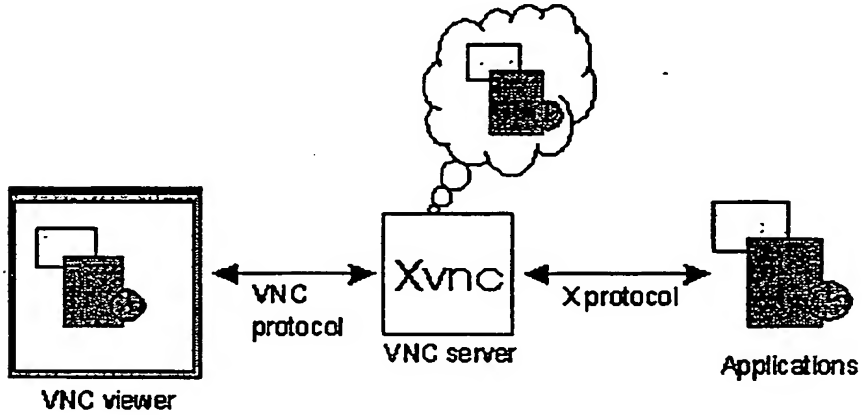
go back to documentation

Home
Screenshots
Why is it free?
Getting started
Documentation
FAQ
Download
Keep in touch
Contribute
Want to help?
VNC people
Search
About us

X-based VNC server

Make sure you've read 'Getting Started' for introductory information.

Xvnc is the Unix VNC server, which is based on a standard X server. Applications can display themselves on it as if it were a normal X display, but they will actually appear on any connected VNC viewers rather than on a physical screen.



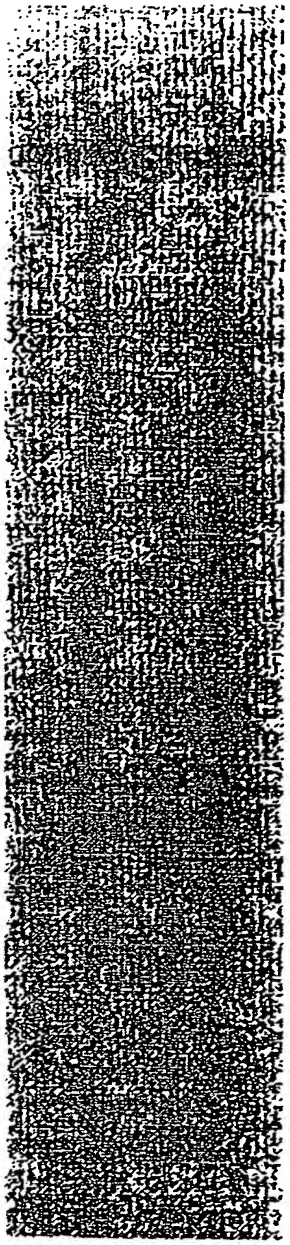
So Xvnc is really two servers in one. To the applications it is an X server, and to the remote VNC users it is a VNC server. By convention we have arranged that the VNC server display number will be the same as the X server display number, which means you can use eg. `snoopy:2` to refer to display 2 on machine 'snoopy' in both the X world and the VNC world.

Normally you will start Xvnc using the `vncserver` script, which is designed to simplify the process, and which is written in Perl. You can edit this to suit your preferences and local conditions. We recommend doing this rather than running Xvnc directly, but Xvnc has essentially the same options as a standard X server, with a few extensions. Running `Xvnc -h` will display a list.

As mentioned in Getting Started, `vncserver` can be run with no options at all. In this case it will choose the first available display number, start Xvnc as that display, and run a couple of basic applications to get you started. You can also specify the display number, in which case it will us that number if it is available and exit if not, eg:

```
vncserver :13
```

Run with the `-help` argument to see the other options. The important ones are as follows:



`-name` name
: Each desktop has a name wh[redacted]ay be displayed by the viewer. It defaults to 'X' but yo[redacted]n change it with this option.

`-geometry` width xheight
: Specify the size of the desktop to be created. By default this will be slightly smaller than your current X display, if set, otherwise 1024x768.

`-depth` depth
: Specify the pixel depth in bits of the desktop to be created. By default this will be the same as your current X display, if set, otherwise 8.

`-pixelformat` format
: Specify pixel format for server to use (BGRnnn or RGBnnn)

In general, you can specify standard X server arguments to `vncserver` and they will be passed through to Xvnc. Again, `Xvnc -help` will list its options, including:.

`-economictranslate`
: The server normally uses a lookup table for translating pixel values when the viewer requests a different format from the native one used by the server. This can use up to 256Kbytes per connected viewer, so if you have many viewers you may wish to specify this option which will save memory at the expense of a little bit of speed. Only relevant for 16-bit-deep desktops.

`-cc` n
: Sets the colour Visual class used by the server. Some X applications don't cope too well with the TrueColor visual normally used by an 8-bit-deep Xvnc. You can make the server use a PseudoColor visual by specifying `-cc 3`.

The script ~/.vnc/xstartup is executed after the server starts. If you want to change the window manager used in your VNC desktop, for example, this is where you should do it.
The server also writes log files in the ~/.vnc directory. These can be useful for tracking down configuration problems and startup errors.

go back to documentation

- WinVNC can now be run from the S[redacted]menu. Alternatively, you can use the Start->Settings->Taskba[redacted]enu to add a shortcut to your Startup group, which will cause WinVNC to be run every time you log in.

After some initial tests, you may wish to run WinVNC as a service. See below for more information.

Using WinVNC

On starting, WinVNC will add a small, green version of the VNC icon to the system task bar. Clicking on this icon with the right mouse button will cause a menu to be displayed, with the following options on it:

- **Properties** - This will cause the Properties dialog to be displayed, allowing the user to change various WinVNC parameters.
- **Kill All Clients** - This will disconnect all currently connected clients from the server.
- **Close** - Shutdown the server.

Moving the mouse over the icon should cause the IP addresses of the local machine to be displayed, if they can be discovered at that time.

You can connect to the server from another machine using a VNC viewer, as described in the Getting Started page.

WinVNC Properties

The following options are available from the Properties dialog.

Incoming Connections

- Accept Socket Connections - The server normally accepts direct, socket-based connections from the vncviewer program. Clearing this tick-box disables direct connection to WinVNC, so that only the CORBA interface used by our internal version may be used to start a connection. (See AT&T internal version info). For the public version, clearing this will disable any incoming connections.
- Display Number - This allows the user to specify the display number which the server will use. There is normally no need to change this from the default of zero.
- Auto - This tick box indicates to WinVNC whether it should use the display number specified in the Display Number box, or whether it should use the first display number not already in use on the server machine.
- Password - Incoming connections must be authenticated to verify that the person connecting is allowed to connect to this machine. This text box allows your password to be specified for authentication.
- Disable Remote Keyboard & Pointer - Any new incoming connections will be able to view the screen but not send any input.

Update Handling

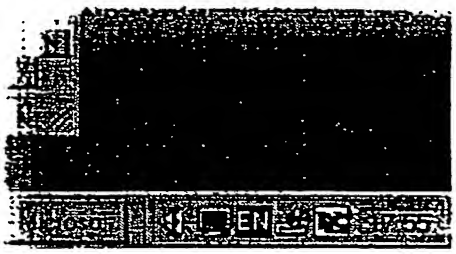
Note that clicking in a window will generally cause it to be updated, so if you have certain applications which don't update very well, try this! The default update handling settings should be the right ones for most people,

Virtual Network Computing

AT&T

Home
Screenshots
Why is it free?
Getting started
Documentation
FAQ
Download
Keep in touch
Contributed
Want to help?
VNC people
Search
About Us

WinVNC - The Windows NT VNC server



WinVNC is a VNC server that will allow you to view your Windows desktop from any VNC viewer. Because Windows in its present, standard incarnation, only supports a single graphical user being logged in at any one time, WinVNC makes the existing desktop of the PC available remotely, rather than creating a separate desktop as happens with the Unix server. It is only fair to emphasise this: VNC does ***not*** make an NT machine into a multi-user server in the same way that Citrix-based software, for example, does. A single NT machine can therefore be accessed by multiple users, but if they all connect at the same time they will all see the same desktop!

On the other hand, WinVNC will run on Windows 95, Windows98, Windows NT 4.0, Windows 2000 and on any future Win32-based systems, without the need to replace any system files or run any OS-specific versions of the program. It is a standard application that can be run from the Start... menu and closed down just as easily.

WinVNC can also be run as a service, which means that you can log in remotely, do some work, and log out again. See below for more details.

And, of course, WinVNC is free. We hope that making the source code available will enable programmers who know more about the internals of Windows than we do to suggest improvements to any and all aspects of WinVNC.

If you haven't yet read the 'Getting Started' page, you might like to do that first to give you the general concepts.

Installation

WinVNC is fairly simple to install and even easier to use:

- Run the WinVNC setup program.
- Optionally, install the default VNCHooks registry settings by selecting Install Default Registry Settings from the WinVNC folder in the Start menu. This will install the default hooks settings, which are tweaked to cope with some common, uncooperative applications, such as the clock. See later for more information about the registry settings.

- WinVNC can now be run from the St[redacted]enu. Alternatively, you can use the Start->Settings->Taskba[redacted]nu to add a shortcut to your Startup group, which will cause WinVNC to be run every time you log in.

After some initial tests, you may wish to run WinVNC as a service. See below for more information.

Using WinVNC

On starting, WinVNC will add a small, green version of the VNC icon to the system task bar. Clicking on this icon with the right mouse button will cause a menu to be displayed, with the following options on it:

- **Properties** - This will cause the Properties dialog to be displayed, allowing the user to change various WinVNC parameters.
- **Kill All Clients** - This will disconnect all currently connected clients from the server.
- **Close** - Shutdown the server.

Moving the mouse over the icon should cause the IP addresses of the local machine to be displayed, if they can be discovered at that time.

You can connect to the server from another machine using a VNC viewer, as described in the Getting Started page.

WinVNC Properties

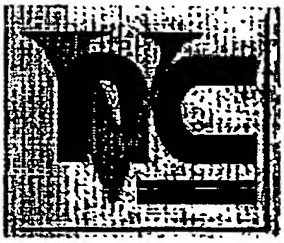
The following options are available from the *Properties* dialog.

Incoming Connections

- Accept Socket Connections - The server normally accepts direct, socket-based connections from the vncviewer program. Clearing this tick-box disables direct connection to WinVNC, so that only the CORBA interface used by our internal version may be used to start a connection. (See AT&T internal version info). For the public version, clearing this will disable any incoming connections.
- Display Number - This allows the user to specify the display number which the server will use. There is normally no need to change this from the default of zero.
- **Auto** - This tick box indicates to WinVNC whether it should use the display number specified in the Display Number box, or whether it should use the first display number not already in use on the server machine.
- Password - Incoming connections must be authenticated to verify that the person connecting is allowed to connect to this machine. This text box allows your password to be specified for authentication.
- Disable Remote Keyboard & Pointer - Any new incoming connections will be able to view the screen but not send any input.

Update Handling

Note that clicking in a window will generally cause it to be updated, so if you have certain applications which don't update very well, try this! The default update handling settings should be the right ones for most people,



Virtual Network Computing

AT&T

Home
Screenshots
Why is it free?
Getting started
Documentation
FAQ
Download
Keep in touch
Contributed
Want to help?
VNC people
Search
About Us

Why are we giving it away?

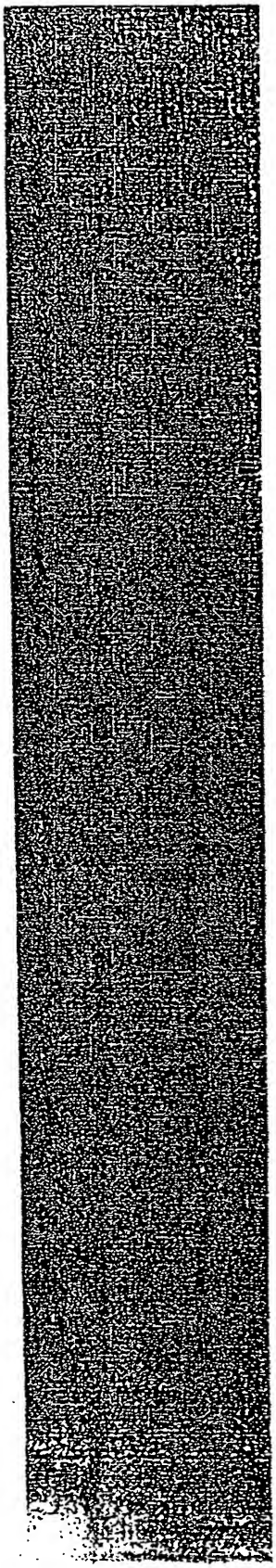
We have been using thin-client systems at our lab for several years and have found them very useful. We hope that there might be others out there who will find VNC useful, port it, find any bugs in it, and give us feedback.

When we made our high-performance CORBA implementation, omniORB, available recently, it was exceedingly popular and so we are releasing VNC on the same terms and conditions: those of the GNU General Public License .

We believe that systems based on stateless endpoints have many advantages, particularly for users who frequently move about within one building or the local area, because of the seamless mobility which comes from maintaining all the state at the server. And with our Java client, these mobile workers can access their email etc from anywhere in the world. But VNC is also useful for the enthusiast at home who has a Windows and a Linux machine but only one monitor.

The initial release contains this documentation, binaries for several platforms, and full source code. It is distributed in the hope that it may be useful but without any warranty, explicit or implied.

For comments, feedback, etc, please see the 'Keeping in touch' page



unless you have applications which caus[illegible] problems.

- Poll Full Screen - Some applications are incompatible with the methods currently used in WinVNC to trap screen updates. For this reason, it is sometimes useful to be able to poll the entire screen in order to check for changes, sacrificing performance for accuracy.
- Poll Foreground Window - Polling only the currently selected window for changes is less CPU intensive than full-screen polling and often gives similar results, for example when using the Command Prompt, which is not normally compatible with WinVNC.
- Poll Window Under Cursor - A variation on Poll Foreground Window, this option causes the window under the mouse cursor to be polled for changes. Both options may be enabled simultaneously if required.
- Poll Console Windows Only - When this option is set, the only windows which will be ever be polled are Command Prompts. This works well in conjunction with Poll Window Under Cursor, to use polling only when the cursor is over a console window.
- Poll On Event Received Only - When this option is set, the screen will only be polled for updates when a mouse or keyboard event is received from the remote client. This is provided for low bandwidth networks, where it may be useful to control how often the screen is polled and changes sent.

The user's settings are saved into the user-specific section of the registry when WinVNC quits, meaning that they will be used next time you run WinVNC.

Running WinVNC as a service

WinVNC can now be made to run as a service process under both Windows NT and Windows 95, by following the instructions outlined below. You can also send Ctrl-Alt-Del to the server, allowing you to unlock a locked workstation, for example, when WinVNC is running as a service on NT. The following 'features' should also be pointed out:

Windows NT 'features':

- There will be one password for the machine, rather than one per user.
- The system tray icon is not always correctly displayed, so you won't always be able to tell whether someone else is connected except by the fact that things work more slowly.

Windows 95 'features':

- Whether or not the VNC password is set per-machine or per-user depends on the settings in the Passwords section of the Control Panel. If Win95 is set to use a different set of registry values for each user then when a user logs in, the password will change from the per-machine VNC password to that user's VNC password. If Win95 is set to use the same settings for all users then the

General features for both Windows NT and Windows 95:

- Anything which causes the Windows VNC server to change screen resolution will also cause all viewers to be disconnected, and you'll need to reconnect.
- When WinVNC is running as a system service, no user-level copies can be run at the same time.

Here's how to get it running as a service, assuming you've already installed it.

1. **Windows NT** : You need to have administrator privileges on the local machine, so log on as administrator if your account doesn't have these.
2. Open a Command Prompt and `cd` into the directory into which you installed WinVNC. eg:

```
D:\> C:
C:\> cd "\Program Files\ORL\VNC"
```

3. Install the WinVNC service using the -install option. A dialog box will appear to indicate the success or failure of the operation.

```
C:\Program Files\ORL\VNC> winvnc -install
```

Windows 95 : The WinVNC service is now running and is installed to run whenever the system boots up into Windows 95.

Windows NT : The WinVNC service is installed and set up to run whenever the machine is booted into Windows NT but IS NOT YET RUNNING ! You can run and stop the WinVNC service using the Windows control panel, or using "net start" and "net stop"

```
C:\> net start winvnc
```

4. If you wish to change the WinVNC settings (eg. password) when it is running as a service and is therefore not visible on the taskbar, you must use the -settings option.
   Open a command prompt and move into the directory into which you installed WinVNC, as described in part 2) above. Use the -settings option to cause the service to display its Properties dialog.

```
C:\Program Files\ORL\VNC> winvnc -settings
```

NOTE : Windows NT : The settings used by the winvnc service are the Default user settings and are stored per-machine, rather than on a per-user basis as is done when running WinVNC normally. Access for all users will be controlled by the one machine-specific password.

NOTE : Windows 95 : If Win95 has been set to use different settings for each user then the settings used are those of the currently logged in user. If no user is logged in or Win95 is set to use the same settings for all users then the settings used are the Default user settings and are stored per-machine, rather than on a per-user basis as is done when running WinVNC normally. (Under Win95, pressing Cancel on the login dialog gives access to the

service from the system, using the "remove option.

Open a command prompt and move into the directory into which you installed WinVNC.

Use the -remove option, which will automatically stop the service and then remove it, and show a dialog box to indicate the success or failure of the operation.

```
C:\Program Files\ORL\VNC> winvnc -remove
```

NOTE : Failure to remove the service usually indicates that it was not installed in the first place!

6. The full command-line options available are as follows (for version R19 and later). You probably won't need anything other than those listed above unless you're a real VNC power-user!

`-run` Causes WinVNC to run normally & ignore rest of command-line.
`-install` Installs the WinVNC service and continues reading the command-line.
`-remove` Removes the WinVNC service and continues reading the command-line.
`-settings` Tells a running copy of WinVNC to show its Properties box.
`-kill` Kills a running copy of WinVNC.
`-about` Tells a running copy of WinVNC to show its About box.

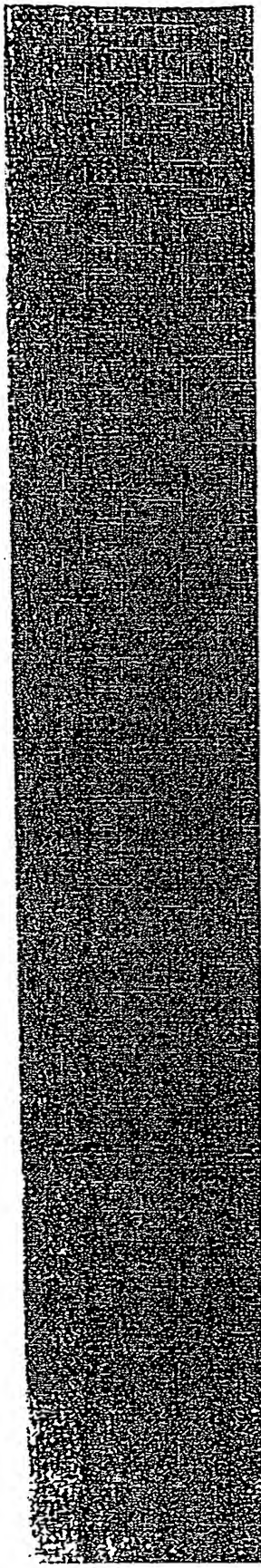
If no options are given then WinVNC runs normally. Multiple option may be given, so, for example, to upgrade from a running copy of WinVNC to a new one, you could use:
`WinVNC_new -remove -install`
which will stop & remove the old copy & install the new one as a service, or
`WinVNC_new -kill -run`
which will stop the running copy & run the new version normally.

WinVNC - Advanced Settings

Extra options have been added to WinVNC for use primarily by system administrators, to tailor the server's behaviour to meet their particular needs. The options are DWORD values which can be set in the system registry, and tools such as the Windows Policy Editor can be used to apply these settings across a large number of machines.

Versions 3.3.2 R5 and later use a more sophisticated organisation of these options to allow more flexibility. It also makes it rather complex, so we're thinking about alternative ways of doing this. WinVNC will currently look for settings in the following places:

1. **Local machine-specific settings**. Options specified here are not overridable. Location:
   HKEY_LOCAL_MACHINE\Software\ORL\WinVNC3\
2. **Local default user settings.** Location:
   HKEY_LOCAL_MACHINE\Software\ORL\WinVNC3\Default
3. **Local per-user settings**. These override the local default user settings. If there is no current user, the username SYSTEM will be



used. Location:
HKEY_LOCAL_MACHINE\Software\[redacted]\WinVNC3\ <username>

4. **Global per-user settings.** These[redacted] only read if AllowProperties has not been set to zero (see below) Location:
HKEY_CURRENT_USER\Software\ORL\WinVNC3

Most options can only be specified in a subset of these places, as specified in each option's description below.

Advanced Options:

AuthRequired
By default, all WinVNC servers will not accept incoming connections unless the server has had its password field set to a non-null value. This restriction was placed to ensure that misconfigured servers would not open security loopholes without the user realising. If a server is only to be used on a secure LAN, however, it may be desirable to forego such checking and allow machines to have a null password. Setting this registry value to zero will disable null-password checking by WinVNC. **Local machine-specific setting.**

AllowLoopback
By default, WinVNC servers disallow any vncviewer connections from the same machine. For testing purposes, or, potentially, when using multiple instances of WinVNC on Windows Terminal Server, this behaviour is undesirable. Setting this registry entry to 1 will cause local-loopback connections to be allowed. Setting it to zero will filter out such connections. **Local machine-specific setting.**

AllowProperties
If this is set to zero, the user is not allowed to view the properties dialog and hence cannot change any settings, including the password. Note that this stops all global per-user settings. A valid password must therefore be in force before using this setting, generally in the local default-user setting. **Local per-user setting.**

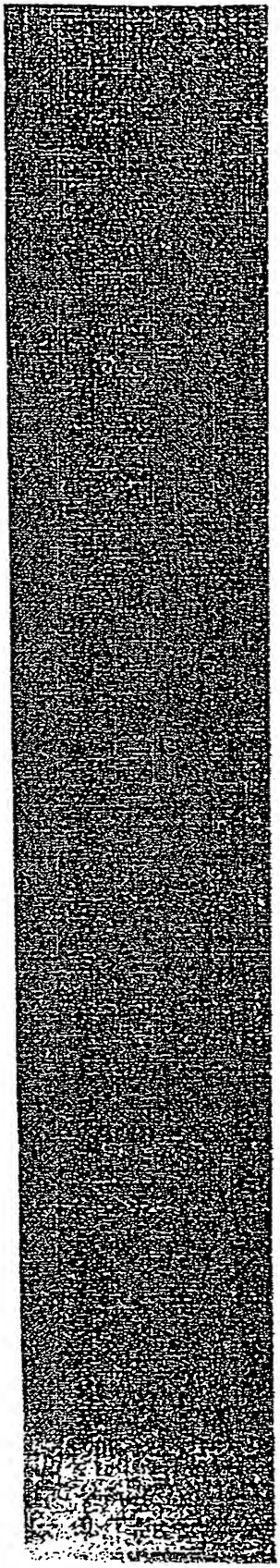
AllowShutdown
If this is set to zero, the user is not allowed to close down WinVNC. **Local per-user setting.**

AutoPortSelect
Causes WinVNC to select the first available display number automatically. Corresponds to the 'Auto' checkbox in the Properties dialog. **Local or Global per-user setting**

CORBAConnect
Only relevant in internal AT&T version. **Local or Global per-user setting**

DebugLevel
DebugMode
Run-time logging of all internal debug messages is now supported. Log data may be output to a file or a console window (or the MSVC debugger if the program was compiled with debugging active.) Two registry keys are used:

DebugMode indicates which logging methods to use,



[1 = MSVC debugger]
2 = Output to log file Winvnc.log in the WinVNC directory
4 = Output to a console window, displayed on-screen
Any combination of the above values may be used. e.g. DebugMode=6 will cause output to be sent both to the WinVNC.log file and to the a console window on the desktop.

DebugLevel indicates how much debug information to present. Any positive integer is valid. Zero indicates that no debugging information should be produced and is the default. A value of around 10-12 will cause full debugging output to be produced.
Local machine-specific setting.

ConnectPriority
By default, all WinVNC servers will disconnect any existing connections when an incoming, non-shared connection is authenticated. This behaviour is undesirable when the server machine is being used as a shared workstation by several users or when remoting a single display to multiple clients for vewing, as in a classroom situation.

ConnectPriority indicates what WinVNC should do when a non-shared connection is received:
0 = Disconnect all existing connections.
1 = Don't disconnect any existing connections.
2 = Refuse the new connection.

This is a **Local machine-specific setting.**

InputsEnabled
Corresponds (inversely) to the 'Disable Remote keyboard and pointer' option in the Properties dialog box. **Local or Global per-user setting**

LockSetting
WinVNC can be made to take actions when a viewer disconnectsby setting this value as follows:
0 - none
1 - lock workstation on disconnect (not currently implemented)
2 - logoff on disconnect
Local or Global per-user setting

Password
Local or Global per-user setting

PollUnderCursor, PollForeground, PollFullScreen, OnlyPollConsole, OnlyPollOnEvent
These correspond to the options in the Properties dialog box. **Local or Global per-user settings**

PortNumber
specifies the port number to be used for VNC
Local or Global per-user setting

SocketConnect
This corresponds to the 'Accept Socket Connections' option in the properties dialog box and is a **Local or Global per-user setting**.

VNCHooks - Advanced Settings

WinVNC uses a special library, VNCHooks, to hook into the other running applications and retrieve notifications of areas of the screen being changed. The VNCHooks library uses the messages sent to visible Windows to decide which areas need considering for update. Not all applications use the same method of updating the screen, so you can tweak the method used by WinVNC for particular applications by editing the registry. All the entries listed can be found under

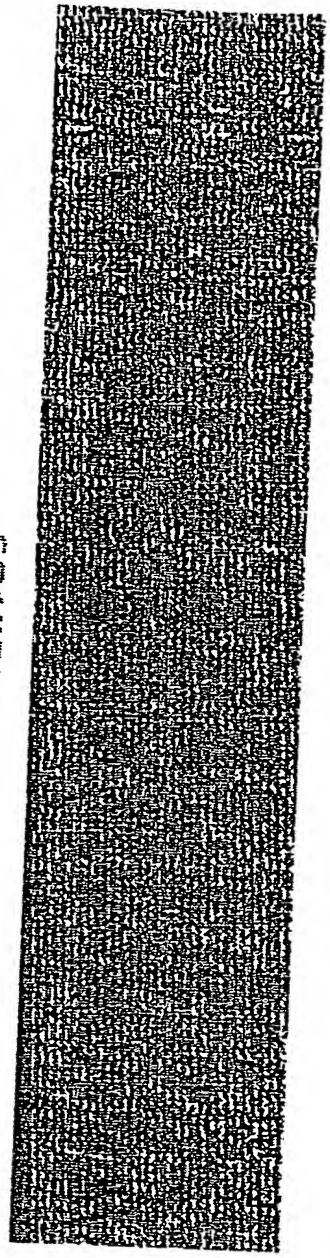
HKEY_CURRENT_USER\Software\ORL\VNCHooks\Application_Prefs

- use_GetUpdateRect
  When a window recieves a message, (WM_PAINT), indicating that it should repaint itself, it is possible to find out precisely which regions have changed, so that WinVNC need only scan those for potential updates,increasing efficiency. However, this can cause graphical glitches occasionally, particularly when an application scrolls the contents of its window, in which case only the revealed section of the window is marked as needing to be updated. If these glitches prove to be a problem then edit the `<appname>\use_GetUpdateRect` entry in the registry. A value of one indicates that this optimisation will be used, while a value of zero indicates that it will not.

- use_Timer
  A number of Windows applications, most notably the Clock program, use WM_TIMER events to trigger updates to their displays, rather than WM_PAINT messages. By default, timer messages are not used to notify WinVNC of potential updates, since many programs use timer events for purposes other than updating the screen. As a result, the clock and a few other applications don't normally update correctly under WinVNC. The fix to this is to edit the `<appname>\use_Timer` entry in the registry. A value of one indicates that WM_TIMER messages will trigger WinVNC updates, while a value of zero indicates that they will not.

- use_KeyPress
  Some Windows applications write characters directly to the screen when a user types into a window, rather than using WM_PAINT messages to cause the text to be redrawn. To fix this, WinVNC can scan the window every time a key is pressed, in order to catch the change. To set this value for a problem application, edit the `<appname>\use_KeyPress` entry in the registry. A value of one indicates that key presses will cause updates, while a value of zero indicates that they will not.

- use_LButtonUp, use_MButtonUp , use_RButtonUp,
  Some Windows applications update the display directly in response to mouse clicks, without using intermediate WM_PAINT messages, for example. In order to catch such updates, it is necessary to trigger WinVNC to update the relevant window whenever the left mouse button is released. To set this value for a problem application, edit the `<appname>\use_LButtonUp` entry in the registry. A value of one indicates that left-button clicks will cause updates, while a value of zero indicates that they will not. The same rules apply to the middle and right buttons using the



appropriate value name.

- use_Deferral
  The VNCHooks library catches messages sent to windows before they are dealt with by the window. As a result, sending an update message to WinVNC to indicate the potential change can result in WinVNC sending the updated area to the client before it has actually been redrawn by the application! This is a common problem, especially on multiprocessor versions of NT, so deferred updates are used by default. Deferred updates are handled by posting a custom message back into the window's own message queue rather than posting to WinVNC directly. By the time this custom message is seen again by the VNCHooks library, the message that caused it will have been handled and the update can then be forwarded to WinVNC without danger of being handled prematurely. A few programs don't handle these extra messages in their queue very well, so this optimisation is optional. It can be set by editing the `<appname>\use_Deferral` entry in the registry. A value of one indicates that deferred updates will be used, while a value of zero indicates that they will not.

Running on other Win32 systems

WinVNC runs fine on NT3.51 but the absence of a system tray means that the Properties dialog cannot be accessed. In addition, Ctrl-Alt-Del from clients cannot be correctly interpreted under NT 3.51, limiting WinVNC's usefulness when run as a service on this platform. It also runs on NT5 beta. If you have a choice we recommend NT4.0 with the latest service packs installed.

Problems?

If you have difficulties which are not covered by this document, try reading the FAQ. If that doesn't help then try the mailing list. If you try to contact the developers directly, please remember that VNC has hundreds of thousands of users, and we cannot, in general, respond to individual queries. We will read your message, but don't expect an answer!

go back to documentation

Virtual Network Computing

AT&T

MacVNC - VNC server for Macintosh

Beta 2 - 7/ 6/99
Alpha 1 .1- 25/1/99
Alpha 1 -16/7/98 .

Use of the Mac server should be straightforward if you are familiar with the concepts behind VNC, and have a suitable VNC viewer running on another machine. Read 'What is VNC? ' and 'Getting Started' if you haven't already done so - they'll give you a good overview, despite being Unix and Windows-orientated.

Note. This is a beta release. As with all VNC software, you use at your own risk! We welcome bug fixes/ comments.

Requirements:

To use the Macintosh VNC server, you will need:

- A version of MacOS greater than 7.6.1,
- A Power PC machine
- Open Transport (1.1.3 or later).

Quick Start

- Drag vncPatches from the ->Extensions folder of the distribution onto a closed System Folder
- Drag the VNC Controls from ->Control Panels onto a closed System Folder
- Restart the machine
- Run VNCServer
- Run VNC Controls and set the password/ desktop name. Set display number to 0.
- Connect with a VNC client to <hostname>:0 or with a Java capable web browser to http://<hostname>:5800

Introduction

The Patches

The installation is in three parts, the patches, the server and the control panel. The patches (in a system extension called vncPatches, stored in the ->Extensions folder of the distribution) catch draws to the screen and pass them on to the server. To install these patches you need to put them in the ->Extensions folder inside your System folder (you can drag the vncPatches

into effect.

The Server

The actual work horse of the installation is the VNCServer program. Note: even though you've installed the patches, no one will be able to access your machine unless the server is running as well. To automatically have the server run at start up you can put it (or one of its aliases) into the Startup Items sub folder of your system folder. Otherwise, just double click it like any other application.

The Control Panel

Once the server is running, you'll need to set the password (the default is none, which means the server will refuse all connections). The server currently supports 3 ways of setting options, via a web browser, the contro panel or Apple Script. The easiest and most reliable way is to use the control panel (called VNC Controls). Note: The server needs to be running (or at least have been run) the first time you use the control panel (why?). Also, the control panel may pause for a few seconds (more on over loaded machines) whilst it communicates with the server. If the control panel is put into the Control Panels folder (inside the System Folder), then it will be accessible by selecting the Options menu option from the server.

The first two options determine how your screen will be described to clients, the desktop name is what external viewers will see in their title bars, while the display number is what they'll have to add to your host name for them to connect properly (i.e. if you set the server running on machine 123.456.789.789 to have a display number of 0, then viewers can connect by specifying 123.456.789.789:0 or computer.room.place.com:0 as the host name). The display number is NOT normally the TCP/IP port number (see web site for further details). The password box is (surprise, surprise) where you should enter your desired password. Only the first 8 characters will be used in authentication but it is case sensitive (i.e. thispassword is the same as thispassXXX but not the same as THISPASS).

Macintoshes have a key (the command key) that most other computers lack, which means that for an external client to send command key events, some sort of key press equivalent needs to be set. The four drop down menus specify which key combination will be used as command. For example, if you wanted your external clients to get command-c by pressing left control + right alt + F3 + c you'd select "L Control", "R Alt", "F3" and "None" from the four menus. Other key presses can be set via the Apple Script interface.

The advanced button on the control panel takes you though to a slightly different set of options, which are described more fully on the panel itself.

If the VNC server preferences file is locked for any reason, no option settings will be allowed through any of the 3 interfaces. Also the web interface and the Apple Script interface can be disabled by setting the appropriate flags on the Advanced settings page.

Once the initial settings have been set, that's it. The server can be stopped by selecting Quit from the file menu or by pressing command- Q.

The Web Interface

capable web browser can connect to http://<host name>:58xx (where xx is the display number) and will be sent a Java viewer that will then act as the VNC client. The Mac server can also accept option changes through local browsers accessing http://<host name>:58xx/settings.html. There is also way to look at the server's internal logs by looking at http://<host name>:58xx/log.html

All the html sent out by the server needs is kept in a directory called Html which should be in the same directory as the VNCServer binary. You can change the look of your server's settings page by changing the contents of this directory. Files ending .tmpl will be put through a template parser before being sent out, so in your customised html you can include things like $HOSTNAME if you want to display your current desktop name. See the included .tmpl files for examples of other variables. You can change where the server looks for its html by changing the HtmlDirectory line in VNC Server preferences (found in the <startup disk>:System Folder:Preferences) or you can include the html/ tmpl files as TEXT resources inside the VNCServer binary. The names of the resources should be the same as the names of the files. Resources are searched before the Html directory.

The web interface can be disabled by removing the Html directory or clearing the "Use Http Options" flag on the Advanced screen of the VNC control panel.

The Apple Script interface

The server can be controlled by Apple Scripts which can be easily written with the Script Editor which comes with every installation of the Mac OS. A sample program is shown below.

```
-- example for communicating with a VNC server
tell application "VNCServer"
	-- set the options to  sensible things
	set password to "password"
	set display number to 34
	set desktop name to "Bob's Computer"
	set cmd equivalent to "ffe3 ffe9 0000 0000"
	-- The following is a very esoteric option described π
	set mouse id to "03045ef2"
	-- get a table of function use (very uninteresting)
	get stats
	-- Clear the previous statistics
	set stats to 0
end tell
```

cmd equivalent

consisting of four space deliminated, unprefixed, four digit hexadecimal numbers (got that?). Seeing as most users are not going to know (or care) what an X key sym is, the control panel includes a way of easily setting this combination via four drop down menus. People who are quite happy with a ctrl/ alt/ shift etc. style key press can use the control panel while others can set what ever key press they like via this Apple Script command.

mouse id

The mouse id option is for people with more than one mouse on their bus who don't want to use the first one as the captured pointing device (for whatever reason). By default, the first pointing device encountered by the CursorDevice manager is the one that is controlled by clients. However, if you want the clients to control the second one, then you're going to have to find out the device ID of the second mouse and pass it to the the server via the mouse id command. The parameter is a string containing the device ID in hexadecimal with no leading characters. Very very few people are going to need this.

Question Index

How Do I....

Q. ... stop other users changing the settings once I have set them?

A. Make the preferences file (which is called VNC Server preferences nd should be in the Preferences file inside the System Folder) read only.

Q. ... use my command key from other machines?

A. The modifiers that correspond to command can be set via the settings page or via Apple Script. The default is Left Alt and Left Control. So L Alt+ L Control + c is the same as command + c. Note: The command key is the one next to the space bar with a propeller symbol or an Apple symbol on it.

Up to four modifiers can be chosen and are described by thier X keysym values. See keysyms.h in the source distribution for more details of X keysyms. The default settings page has drop down menus to choose from a number of common modifiers. To add more modifiers to these menus put

The modifiers can be set from AppleScript like this:

```
tell application "VNCserver"
  set cmd equivalent to "ffe3 ffe9 0000 0000"
end tell
```

The string is four 4 digit hexadecimal numbers describing the modifiers requested. (ffe3 ffe9 correspond to the default L Alt + L Control).

Note: Setting the modifiers to anything strange will produce "Current Modifiers: unknown" on the settings page. Do not panic. This just means that the server doesn't know how to describe the keysyms you've given it. They will still work, it's just the server doesn't have name for them.

Q. ... customise my settings/ logs / initial page?

A. The HTTP server look at the following places for HTTP files:

- The directory specified in the Html directory line in the preference file. (Default is the directory called Html in the same directory as the server)
- Resources of type 'TEXT' inside VNCServer
- Resources of type ' WWW' inside VNCServer (There's a space before the first W in ' WWW')
- Resources of type 'JAVA inside VNCServer

You can therfore override the internal html files by putting new files in the Html directory.
Also, the server will transform certain filenames under cetain conditior

Condition...	Maps ...	... to ...
Connection is local and the server unlocked	settings.html	settings.tmpl
Connection is not local and the server is unlocked	settings.html	localonly.tmpl
Server is locked	settings.html	noserver.html
Server is running properly	index.html or /	javastart.tmpl
Server is not running properly	index.html or /	noserver.html
Status info is available.	log.html	log.tmpl
Status info is not availble	log.html	noserver.html
Status info is available.	log<X>.html	single_log.tmpl
Status info is not availble	log<X>.html	single_log.tmpl

The <X> in the last two files is an ascii character from 0 onwards that describes which log number you want.
Log 0 is the main program log, log 1 is the http log, log 2 is a description of status of the server and a list of active connections, logs 3 and onward are the logs associated with the active connections. The logs are served

Most of the html files served are passed [redacted]gh a template parser before being sent to the client. The template parser translates some tokens into useful information:

Token ...	Expands to...
$SETTINGS_LINK	<A href=<host>/settings.html>Settings</A> if the connection is local and the server unlocked. Nothiong otherwise.
$HOST	The ip address of the host.
$WIDTH,$APPLETWIDTH	The width of the screen
$HEIGHT,$APPLETHEIGHT	The height of the screen
$LOG<X>	The specified log. <X> is an ascii character starting at 0 which specifys the log
$MAINPAGE	A fully formed URL to the index page
$PORT	The port number for vnc connections (normally 5900)
$DESKTOP	The user specified name of the desktop
$DISPLAY	The display number. Normally 0
$MODIFIERS	English description of the modifers that correspond to the command key
$THELOG	A log. This token only works in single_log.html, which is only used when the user requests log<X>.html. See above for what the <X> means.

Settings can be changed by sending a POST event to the server, via a an HTML form.
The labels in the POST correspond to various variables:

Label...	Variable ...
PASS	The password
PAS2	The password again. It will be set if both of these are the same
DISP	Display number
DESK	Desktop name
Key1,Key2,Key3,Key4	The command key modifiers.

In response to a POST event (usually sent from the settings page) the server returns changed.tmpl unless the password was incorrectly set in which case passwrong.tmpl is sent.

Q. ... paste into my Mac from another machine?

A. The short answer is "You can't". The longer answer is "You can't because of the slightly odd way the Mac handles clipboards". This problem is being currently being investigated. Pasting from the Mac to other machine should work.

Q. ... communicate with the developers?

A. We run a VNC mailing list run that has a pretty large membership list and is read by all the VNC developers (not just the Mac one), so any questions you have can probably be answered by posting to this list. To join the list go here. Note that asking questions which are answered in this document or in the FAQ is a Bad Idea.

Why?..

Q. ... does my machine keep crashing?

A. VNCServer, in its fastest mode, breaks a general guideline in that it patches ShowCursor (which has to be interrupt time safe) with some code that isn't interrupt safe. This may sometimes cause bad things to happen. If this is happening to you, you can switch to an interrupt-safe version of ShowCursor by changing the "Use Error prone updates" flag in the security resource. See above for details. The downside is that updates that happen when the mouse button is down (like drags and menu operation) may not work so well.
As mentioned above, there are also some known problems with the hextile encoding. You should find that the server is more reliable if you switch this off.

Q. ... doesn't the File/Options(HTTP)... menu item work?

A. The server options are set using a web browser via an HTTP connection. If your machine isn't set up for this already, you need Internet Config to make this work. From the IC faq:

> Each new release of Internet Config is posted to MacGifts and is therefore available Info-Mac and UMich (and their mirror sites). It is also available from any site that holds Peter Lewis' software, namely AOL, AMUG, PopCo, Australia, Japan, and Switzerland.
>
> Internet Config is also available as a NewsWatcher helper and on the TidBITS site.
>
> Finally, all sorts of IC related material, including the latest version of the IC software, is available from the IC home sites in Australia and the USA.

Q: ... won't the control panel run unless the server has been run at least once before?

A: The reason for this is that the panel either communicates directly with the server (through Apple Script) or alters its preferences file. If the server has never been run before, there will be no preferences file on the system and the panel won't be able to do anything.

Any comments regarding this program should be directed to the VNC mailing list.

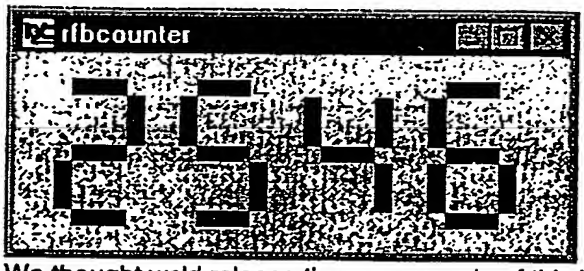
go back to documentation

Virtual Network Computing

AT&T

rfbcounter

A really simple VNC server.



We thought we'd release the source code of this just to show that a VNC server doesn't have to be big and complicated, and that you can display things other than desktops!
Rfbcounter creates and updates a simple numeric counter. It does so by generating raw VNC protocol to produce the digits, and does not have a framebuffer of its own.

The name comes from the fact that the VNC protocol is sometimes referred to as the RFB protocol. RFB stands for 'Remote Framebuffer'.

The source code for rfbcounter can be found, tarred and gzipped, here. It is a single C file with a couple of headers, and should compile easily on Linux, and with very little effort on Win32 and other flavours of Unix.

Usage:

```
rfbcounter [-clock] display-number width [fg] [bg]
```

eg. if you run

```
rfbcounter 5 300
```

you can then connect a VNC viewer to display 5 on that machine and you'll get a counter 500 pixels wide. The fg & bg arguments are single byte values specifying foreground and background colours in BGR233. By prefixing with a zero, you can use octal to simplify color selection, for example 0007 0100 is bright red on a dark blue background. If you specify the -clock option, rfbcounter will display the current time instead of a free-running counter, and update it once a minute.

Note: rfbcounter does not do any pixel translation. Your viewer *must* cope with 8-bit bgr233 format for this to work. On Unix, use the -bgr233 option to vncviewer.

This is not an example of good VNC code. We ignore lots of messages

Virtual Network Computing

VNCviewer for X

Run it and give the display as an argument:

```
vncviewer snoopy:2
```

where 'snoopy' is the name of the machine, and '2' is the display number of the VNC server on that machine. The `-h` argument will show you the other options. The important ones are as follows:

`-shared`
: When you make a connection to a VNC server, all other existing connections are normally closed. This option requests that they be left open, allowing you to share the desktop with someone already using it.

`-display` Xdisplay
: This allows you to specify the X display on which the VNCviewer window should appear.

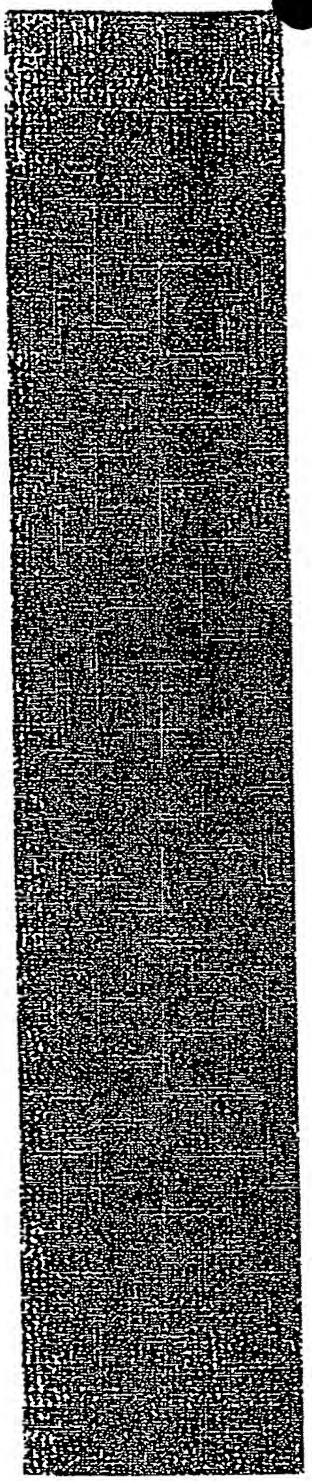
`-passwd` password-file
: If you are on a filesystem which gives you access to the password file used by the server, you can specify it here to avoid typing it in.

`-viewonly`
: Specifies that no keyboard or mouse events should be sent to the server. Useful if you want to view a desktop without interfering; often needs to be combined with `-shared`

`-geometry` geometry
: Standard X position and sizing specification.

`-bgr233`
: This tells the VNC server to send pixels which are only 8 bits deep. If your server desktop is deeper than this then it will translate the pixels before sending them. Less data will generally be sent over then network, which can be a big advantage on slow links, but the server may have to work a bit harder, and you may get some colour mismatches. This is the 8-bit true colour pixel format used by the java client, with the most significant two bits of each byte representing the blue component, the next three bits representing green and the least significant three representing red, hence 'bgr233'.

`-raw, -copyrect, -rre, -corre, -hextile, -nocopyrect, -norre, -nocorre, -nohextile`
: These options affect which encodings vncviewer tells the VNC server it can cope with. Normally it requests CopyRect, Hextile, CoRRE and RRE in that order. The options alter this behaviour in the obvious way, e.g. specifying just `-raw` means Raw will be requested before any of the others,



`-depth d`
This is only useful on a (real) X server which supports multiple depths. On such a display vncviewer will try to find a Visual of the given depth. If successful this means that the appropriate pixel format will be requested from the VNC server. You cannot use this to force a particular depth from the VNC server. The only option which does this is -bgr233.

`-truecolour`
vncviewer will try to find an X visual of the TrueColor class.

`-owncmap`
vncviewer will try to find a PseudoColor visual and use its own Colormap.

`-period ms`
This tells vncviewer not to request incremental framebuffer updates more often than the given period in milliseconds. If you have a very fast client and server, you can use this option to limit the rate of updates - this can result in using less network bandwidth.

`-wmdecoration wxh`
Normally the viewer window will be the same size as the desktop you're connecting to, plus any decorations added by your local window manager. If your local screen is too small to display this then it will make the window as big as it can while still allowing room for decorations, and display scrollbars. This option allows you to specify how big your window manager's decorations are, so that it can make this decision correctly. In particular, some people run without a local window manager, and specifying -wmdecoration 0x0 will then allow a completely full-screen window.

`-rawdelay ms`
This is useful for checking exactly which parts of the screen are being updated. For each update rectangle vncviewer puts up a black rectangle for the given time before putting up the pixel data. This only highlights pixel data sent using the raw encoding.

`-copyrectdelay ms`
This works as with -rawdelay above, but highlights the areas copied using the copyrect encoding.

`-debug`
This prints out all the data received from the VNC server in both hex and ASCII.

`-listen`
This is used for AT&T's internal version of VNC. It causes vncviewer to listen on port 5500+<display-number> for reverse connections from a VNC server. See http://www.uk.research.att.com/vnc/internalversion.html

go back to documentation

Virtual Network Computing

AT&T

VNCviewer for Windows

You can run the windows vncviewer from the command line or from a shortcut and it will prompt you for a display:

```
vncviewer
```

You can specify a display on the command line:

```
vncviewer snoopy:2
```

And you can run it with -h to get a list of other important options. The full list is below. These can all take either - or / as the switch character. Most of the options can also be set from the 'Options...' dialog box which is available from the initial connection prompt before connecting, and some from the system menu by clicking the VNC logo in the top-left corner of the window after connection, and selecting 'Connection options...'.

The system menu also allows you to see some information about the connection, start new connections, and send a Ctrl-Alt-Del to a remote machine. This will only have an effect if the remote server is able to interpret it, currently only true for WinVNC running as a service under NT 4.

Keystrokes such as Ctrl-Esc and Alt-Tab may be interpreted at the local (viewer) machine. If you want to send them to the remote machine, you can use the options on the viewer menu to send individual Ctrl-down, Ctrl-up, Alt-down and Alt-up keystrokes. For example, to type Ctrl-Esc on the remote machine, send Ctrl-down using the menu, press Esc, and then send Ctrl-up (or just tap the Ctrl key) to release the Ctrl key at the remote end.

Command line options:

`-shared`
: When you make a connection to a VNC server, all other existing connections are normally closed. This option requests that they be left open, allowing you to share the desktop with someone already using it.

`-8bit`
: The viewer will normally accept whatever pixel format the server offers and do the translation locally. This forces it to request 8-bit true-colour (BGR233) from the server, which will reduce network traffic. Useful over modems.

`-emulate3`
: Users with a two-button mouse can emulate a middle button by

command line or in the dialog box.

`-swapmouse`

This option was more commonly used before the 3-button emulation was available. Normally the PC buttons left-middle-right are mapped on to X buttons 1,2,3. This switch causes them to be mapped onto buttons 1,3,2, which may be more useful for two-button users who only have left-right, because they will then get buttons 1 & 2 instead of 1 & 3. If combined with 3-button emulation, this also causes the middle button to emulate button 3 instead of button 2. This may be useful if you use button 2 more.

`-emulate3timeout`

When using 3-button emulation, both mouse buttons must be pressed within a certain period for them to be registered as a single middle-click instead of separate left and right clicks. This option allows that time period to be specified in msec. The default is 100.

`-emulate3fuzz`

When using 3-button emulation, both mouse buttons must be pressed within a certain distance of each other for them to be registered as a single middle-click. This option allows that distance to be specified in pixels. The default is 4.

`-fullscreen`

This causes connections to start in full-screen mode by default. See below for more details.

`-disableclipboard`

Clipboard changes caused by cutting or copying at either the viewer or server end are normally transmitted to the other end. This option disables clipboard transfers.

`-belldeiconify`

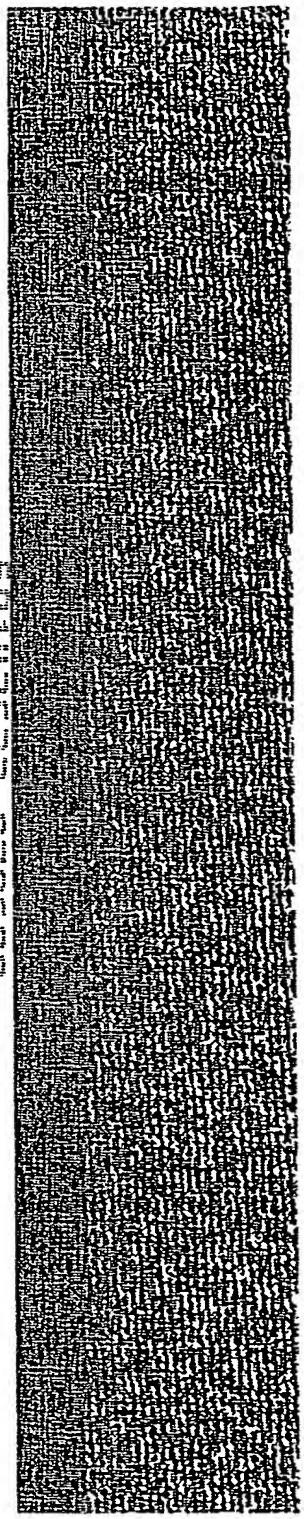
VNC allows for the transmission of a 'bell' character, causing a beep at the viewer if it has sound facilities. You can set the sound to be used for the bell under the VNCviewer section of 'Sounds' in the Control Panel. Often a beep will happen because you are being notified of something such as email arriving or compilation finishing. This switch causes a minimized vncviewer to be un-minimized when a bell character is received.

`-listen`

In the internal version of VNC used at AT&T Laboratories Cambridge, the server can initiate connections to the clients under CORBA control. This switch puts vncviewer into listening mode where it can accept these connections, but it also has a useful side-effect which may be of interest to those outside AT&T using the public version. A listening vncviewer does not pop up a connection dialog, but instead installs itself in the system tray. From there you can easily start up new connections and can set default options to be used for them during this instance of the program. **RECENT NEWS!** The latest versions of WinVNC can initiate the connection to a viewer using the 'Add New Client' menu option. For this to work, the viewer must be in listening mode.

`-keyboard XKBname`

Windows uses an internal and not very helpful name for the keyboard layout currently selected for an application. You can



specify it on the command line to cause vncviewer to attempt to load this in the future. Note that vncviewer does not currently support 'dead keys', and that the differences between language and keyboard are confusing and the way they are handled is different in Windows 95 and NT. But this may help a bit.

`-logfile filename`
VNCviewer (R6 and later) has a logging mechanism which can save some debugging information to a file or display it on a console. This option specifies the name of a file to which a log will be written.

`-loglevel n`
This option controls the amount of logging information sent to the log file. The default is zero, and higher values (up to about 12) will provide more detail.

`-console`
In addition to, or instead of, logging to a file, this option will cause the debugging information to be sent to a console window.

`-viewonly`
In View-only mode, no mouse or keyboard events will be sent back to the server. This is useful for teaching sessions or other situations where you want to observe but don't want to interfere.

`-restricted`
In restricted mode, most of the items are removed from the menu, so that the user cannot, for example, send a Ctrl-Alt-Del to the remote end.

Full-screen mode

Vncviewer can now be switched into a fullscreen mode. This is particularly useful when connecting to a remote screen which is the same size as your local one. If the remote screen is bigger, you won't get any scrollbars, but you can scroll by bumping the mouse against the edge of the screen.

To leave fullscreen mode you must disable it from the menu, but the menu is no longer visible! So you have to bring the taskbar to the front by typing Ctrl-Esc Esc, and then right-click on the vncviewer icon. A dialog box will appear when you select fullscreen mode to remind you of this; if, after a while you get annoyed with the dialog box, you can disable it by creating a DWORD registry value named
`HKEY_CURRENT_USER\Software\ORL\VNCviewer\Settings\SkipFullScreenPrompt`

and setting it to 1. A simpler method will be in a future version!

You can cause new connections to start in fullscreen mode using the -fullscreen command-line option.

See also 'What's new in the Windows VNC package?

go back to documentation

Virtual Network Computing

AT&T

VNCviewer for Java

Because Java applets can only make connections back to the machine from which they were served, each of the VNC servers actually incorporates a small web server. This runs on port 58xx, where xx is the display number, and will only serve the Java applet classes and an HTML page which contains them.

This means that you should be able to point any Java-capable browser at, for example:

```
http://snoopy:5802/
```

and you should, after a short pause, be able to connect to your VNC session.

If you are using the X-based VNC server, you may need to specify the directory which contains the class files in the vncserver script. The Win32 server has the classes embedded in the server itself.

Java implementations seem to vary widely both in how fast they can read from the network and how fast they can draw to the screen. It's worth using the Options dialog to experiment with different encoding schemes for any given network and browser.

Running as an application

You can run the viewer outside a browser using, for example:

```
java vncviewer HOST snoopy PORT 5902
```

Note that you need to specify the actual VNC port number and not the display number or the HTTP port number here! The precise command line will depend on your particular Java installation.

go back to documentation

VNC Viewer for Macintosh

Beta 2 -7/6/99
Beta 1 - 16/7/98 .

Note. This is a beta release. As with all VNC software, you use at your own risk! We welcome bugfixes/ comments.

Requirements:

To use the Macintosh VNCviewer, you will need:

- MacOS 7.1 or greater,
- the Threads Manager,
- and Open Transport (1.1.1 or later) or MacTCP, though OT will work much better.

Introduction:

Use of the Mac viewer should be straightforward if you are familiar with the concepts behind VNC, and have a suitable VNC server running on another machine. Read 'What is VNC? ' and 'Getting Started' on the VNC web site if you haven't already done so - they'll give you a good overview, despite being Unix and Windows-orientated.

On starting the program, you are presented with a dialog box requesting the server name and display number. Type, eg. 'snoopy:0', or select a recent connection from the pull-down list. You can type a dotted IP address in place of the name, eg: '192.168.1.3:0'. Note the display number is NOT normally the TCP/IP port number. You can also pop up a list of options, the important ones are:

- **Share desktop**
  When you make a connection to a VNC server, all other existing connections are normally closed. This option requests that they be left open, allowing you to share the desktop with someone already using it.
- **Allow only 8-bit encoding**
  This forces the viewer to request simple 8-bit true-colour (BGR233) from the server regardless of local or remote pixel depth, which can reduce network traffic - useful over modems.
- **View only**
  In View-only mode, no mouse or keyboard events will be sent

- **Scale desktop to window**
  This scales the remote display to fit the local window. This is a bit rough at present, but may be useful in certain circumstances.

If the connection is successful, you will be asked for your password, after which the remote desktop should appear. The File.. menu will allow you to start new sessions. The Connections menu lists open connections, along with icons representing their current status. See the 'Symbol Reference' option on the Apple menu for a description of these icons.

Mouse Buttons (new for Beta 2)

As the Mac mouse (usually) only has one button, some mechanism is needed to transmit left and middle combinations. Pressing command + n will toggle the use of button n. The current status of the button toggles is shown on the title bar of each connection. Initially the only toggle set is button 1. This means when the mouse is clicked a button 1 click event is sent. If you wanted to send a button 2 click you would turn off button 1 (via command + 1) then turn on button 2 (via command + 2) then click the mouse. To return to normal (button 1) use you'd need to untoggle button 2 (command + 2) and retoggle button 1. In this way all 8 combinations of 3 button superpositions can be sent. Admittedly this is a bit long winded if you only want to send a button 3 click (as you would for pasting in Unix) so command + 4 and command + 5 are set to send a single click for buttons 2 and 3 respectivly.

So, to summarise,

Command + 1	Toggle use of button 1
Commnad + 2	Toggle use of button 2
Command + 3	Toggle use of button 3
Command + 4	Send single button 2 click
Command + 5	Send single button 3 click

Other modifier keys will work as normal so contol shift or alt dragging can be done with any combination of mouse buttons.

Other features:

Keeping your finger on the Propeller key (sometimes called the Command key) when starting off brings up the extra preferences box.

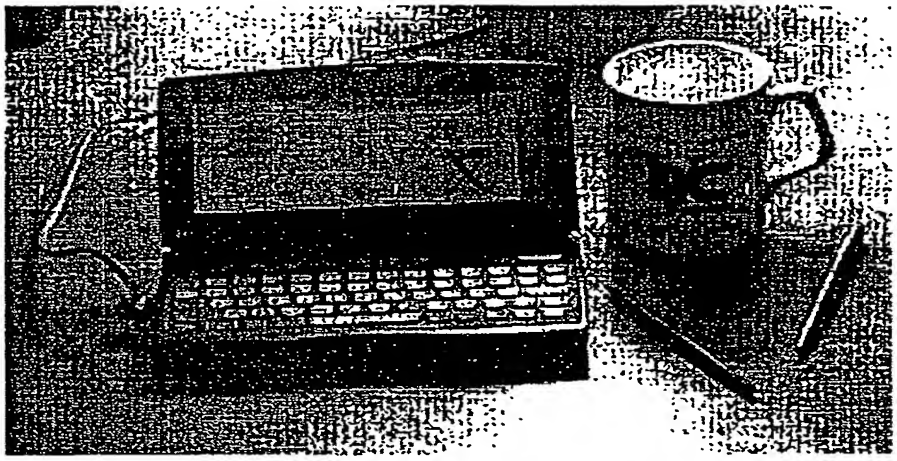
- Enable Logging - will trigger the production of log files
- Kill Preferences - Delete the preferences file and prevent this session's preferences being recorded. The only thing currently stored in the preferences file is the list of recent connections.

Known Issues:

The "Scale to window" option is not pixel perfect, causing some redraws to be offset by one pixel. This causes a ripple effect as the

Home
Screenshots
Why is it free?
Getting started
Documentation
FAQ
Download
Keep in touch
Contributed
Want to help?
VNC people
Search
About Us

VNCviewer for Windows CE 2.x

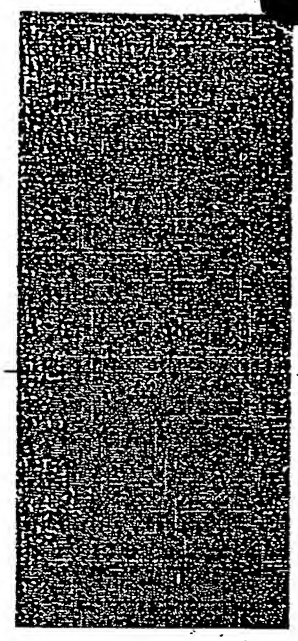


Now you can view X Windows applications on your PDA!

The VNC viewer for Windows CE is based heavily on the Win32 viewer. It requires Windows CE 2.0 or later.

To install, make sure that TCP/IP is working correctly on your WinCE machine (try browsing the web), and we suggest you set the communications link to the fastest speed it can manage. Try connecting to the smallest desktop you can, initially. If you're viewing a PC or Mac, set the server's screen to 640x480 or so and 256 colours. If you're connecting to an Xvnc server. then you can of course create a desktop of any size you like. Larger screens will work - I've viewed a 1280x1024x24 screen here, but the initial screen download takes quite a while, and in this version a copy of the entire remote screen is kept in the PDA's memory, so you'll need plenty of RAM free!

Copy the appropriate vncview.exe binary onto the PDA and run it. If you have used the Windows viewer the dialogs and menus should look very familiar. You might want to use the Options button in the Connection dialog to request 8-bit pixels only. You can set this from the command line, so I suggest you create a shortcut to the vncview executable and include the '-8bit' option in the target command line.
Most of the command-line arguments and options are taken directly from the Win32 viewer, and you should look at the Windows viewer documentation for more details, but there are a couple of CE-specific options:

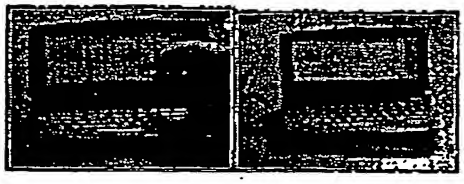


These options allow you to specify the layout to be used

`-slow`

We have recently improved the speed dramatically by writing directly to a DIB instead of using CE's very slow GDI calls. If this causes problems, or if you just want to see what it used to be like, you can try this switch!

The Windows CE viewer does not operate in full-screen or listening mode. There is no console-mode debugging, and some of the other Win32 viewer options are not implemented.



This is certainly the smallest X terminal I've ever used!

go back to documentation

Virtual Network Computing

AT&T

- Home
- Screenshots
- Why is it free?
- Getting started
- Documentation
- FAQ
- Download
- Keep in touch
- Contributed
- Want to help?
- VNC people
- Search
- About Us

The VNC protocol

The VNC system is based on the concept of a remote framebuffer or RFB. We therefore often refer to the protocol underlying VNC as the RFB protocol .

- The protocol specification (Acrobat format)
- The header file rfbproto.h which is used by the system.(Acrobat format)

go back to documentation

The RFB Protocol

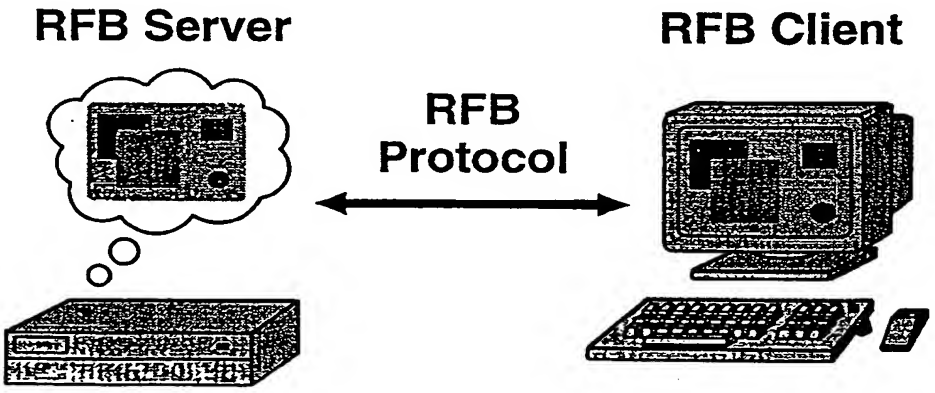
Tristan Richardson
Kenneth R. Wood
ORL
Cambridge

Version 3.3
January 1998
Revised 16 July 1998

1 Introduction

RFB (*"remote framebuffer"*) is a simple protocol for remote access to graphical user interfaces. Because it works at the framebuffer level it is applicable to all windowing systems and applications, including X11, Windows 3.1/95/NT and Macintosh.

The remote endpoint where the user sits (i.e. the display plus keyboard and/or pointer) is called the RFB client. The endpoint where changes to the framebuffer originate (i.e. the windowing system and applications) is known as the RFB server.



RFB is truly a "thin client" protocol. The emphasis in the design of the RFB protocol is to make very few requirements of the client. In this way, clients can run on the widest range of hardware, and the task of implementing a client is made as simple as possible.

The protocol also makes the client stateless. If a client disconnects from a given server and subsequently reconnects to that same server, the state of the user interface is preserved. Furthermore, a different client endpoint can be used to connect to the same RFB server. At the new endpoint, the user will see exactly the same graphical user interface as at the original endpoint. In effect, the interface to the user's applications becomes completely mobile. Wherever suitable network connectivity exists, the user can access their own personal applications, and the state of these applications is preserved between accesses from different locations. This provides the user with a familiar, uniform view of the computing infrastructure wherever they go.

2 Display Protocol

The display side of the protocol is based around a single graphics primitive: *"put a rectangle of pixel data at a given x,y position"*. At first glance this might seem an inefficient way of drawing many user interface components. However, allowing various different encodings for the pixel data gives us a large degree of flexibility in how to trade off various parameters such as network bandwidth, client drawing speed and server processing speed.

A sequence of these rectangles makes a *framebuffer update* (or simply *update*). An update represents a change from one valid framebuffer state to another, so in some ways is similar to a frame of video. The rectangles in an update are usually disjoint but this is not necessarily the case.

The update protocol is demand-driven by the client. That is, an update is only sent from the server to the client in response to an explicit request from the client. This gives the protocol an adaptive quality. The slower the client and the network are, the lower the rate of updates becomes. With typical applications, changes to the same area of the framebuffer tend to happen soon after one another. With a slow client and/or network, transient states of the framebuffer can be ignored, resulting in less network traffic and less drawing for the client.

3 Input Protocol

The input side of the protocol is based on a standard workstation model of a keyboard and multi-button pointing device. Input events are simply sent to the server by the client whenever the user presses a key or pointer button, or whenever the pointing device is moved. These input events can also be synthesised from other non-standard I/O devices. For example, a pen-based handwriting recognition engine might generate keyboard events.

4 Representation of pixel data

Initial interaction between the RFB client and server involves a negotiation of the *format* and *encoding* with which pixel data will be sent. This negotiation has been de-

signed to make the job of the client as easy as possible. The bottom line is that the server must always be able to supply pixel data in the form the client wants. However if the client is able to cope equally with several different formats or encodings, it may choose one which is easier for the server to produce.

Pixel *format* refers to the representation of individual colours by pixel values. The most common pixel formats are 24-bit or 16-bit "true colour", where bit-fields within the pixel value translate directly to red, green and blue intensities, and 8-bit "colour map" where an arbitrary mapping can be used to translate from pixel values to the RGB intensities.

Encoding refers to how a rectangle of pixel data will be sent on the wire. Every rectangle of pixel data is prefixed by a header giving the X,Y position of the rectangle on the screen, the width and height of the rectangle, and an *encoding type* which specifies the encoding of the pixel data. The data itself then follows using the specified encoding.

The protocol can be extended by adding new encoding types. The encoding types defined at present are *raw* encoding, *copy rectangle* encoding, *RRE (rise-and-run-length)* encoding, *CoRRE (Compact RRE)* encoding and *hextile* encoding. In practice we normally use only the *hextile* and *copy rectangle* encodings since they provide the best compression for typical desktops. Other examples of possible encodings include JPEG for still images and MPEG for efficient transmission of moving images.

4.1 Raw encoding

The simplest encoding type is raw pixel data. In this case the data consists of n pixel values where n is the width times the height of the rectangle. The values simply represent each pixel in left-to-right scanline order. All RFB clients must be able to cope with pixel data in this raw encoding, and RFB servers should only produce raw encoding unless the client specifically asks for some other encoding type.

4.2 Copy Rectangle encoding

The *copy rectangle* encoding is a very simple and efficient encoding which can be used when the client already has the same pixel data elsewhere in its framebuffer. The encoding on the wire simply consists of an X,Y coordinate. This gives a position in the framebuffer from which the client can copy the rectangle of pixel data. This can be used in a variety of situations, the most obvious of which are when the user moves a window across the screen, and when the contents of a window are scrolled. A less obvious use is for optimising drawing of text or other repeating patterns. An intelligent server may be able to send a pattern explicitly only once, and knowing the previous position of the pattern in the framebuffer, send subsequent occurrences of the same pattern using the *copy rectangle* encoding.

4.3 RRE encoding

RRE stands for *rise-and-run-length encoding* and as its name implies, it is essentially a two-dimensional analogue of run-length encoding. RRE-encoded rectangles are not only compressed to the same degree or better than is possible with run-length encoding but, more importantly, they arrive at the client in a form which can be rendered immediately and efficiently by the simplest of graphics engines. Thus, RRE is aimed at situations where compression is desired but the RFB client is insufficiently powerful to perform any decompression fast enough to maintain interactive performance.

The basic idea behind RRE is the partitioning of a rectangle of pixel data into rectangular subregions (subrectangles) each of which consists of pixels of a single value and the union of which comprises the original rectangular region. The near-optimal partition of a given rectangle into such subrectangles is relatively easy to compute.

The resulting encoding on the wire consists of a background pixel value, V_b (typically the most prevalent pixel value in the rectangle) and a count N, followed by a list of N subrectangles, each of which consists of a tuple $< v, x, y, w, h >$ where $v (\neq V_b)$ is the pixel value, (x, y) are the coordinates of the subrectangle relative to the top-left corner of the rectangle, and (w, h) are the width and height of the subrectangle. The client can render the original rectangle by drawing a filled rectangle of the background pixel value and then drawing a filled rectangle corresponding to each subrectangle.

4.4 CoRRE encoding

CoRRE is a variant of RRE, where we guarantee that the largest rectangle sent is no more than 255x255 pixels. A server which wants to send a rectangle larger than this simply splits it up and sends several smaller RFB rectangles. Within each of these smaller rectangles, a single byte can then be used to represent the dimensions of the subrectangles. For a typical desktop, this results in better compression than RRE. In fact, the best compression is achieved when we limit the rectangle size even more - current implementations use a maximum of 48x48. This is because rectangles which do not encode well (typically those containing image data) are sent as raw, while the ones which do encode well are sent as CoRRE. The smaller the maximum rectangle size, the finer the granularity of this decision. With RRE, the whole original rectangle must either be sent as RRE, or the whole thing sent as raw. However, since there is a certain overhead incurred by each RFB rectangle, making the maximum rectangle size too small (and thus increasing the number of RFB rectangles), results in worse compression.

4.5 Hextile encoding

Hextile is a variation on the CoRRE idea. Rectangles are split up into 16x16 *tiles*, allowing the dimensions of the subrectangles to be specified in 4 bits each, 16 bits in total. Unlike CoRRE, tiles are not top-level RFB rectangles. When splitting the original rectangle into tiles this is done in a predetermined way. This means that the position and size of each tile do not have to be explicitly specified - the encoded contents of the tiles

simply follow one another in the predetermined order. The ordering of tiles that we use is starting at the top left going in left-to-right, top-to-bottom order. If the width of the whole rectangle is not an exact multiple of 16 then the width of the last tile in each row will be correspondingly smaller. Similarly if the height of the whole rectangle is not an exact multiple of 16 then the height of each tile in the final row will also be smaller.

Each tile is either encoded as raw pixel data, or as a variation on RRE - this is specified by a type byte for each tile. Each tile has a background pixel value, as before. However, the background pixel value does not need to be explicitly specified for a given tile if it is the same as the background of the previous tile. If all of the subrectangles of a tile have the same pixel value, this can be specified once as a foreground pixel value for the whole tile. As with the background, the foreground pixel value can be left unspecified, meaning it is carried over from the previous tile.

5 Protocol Messages

The RFB protocol can operate over any reliable transport, either byte-stream or message-based. There are two stages to the protocol; an initial handshaking phase followed by the normal protocol interaction.

The initial handshaking consists of *ProtocolVersion*, *Authentication*, *ClientInitialisation* and *ServerInitialisation* messages, as described below. Note that both client and server send a *ProtocolVersion* message.

The protocol proceeds to the normal interaction stage after the *ServerInitialisation* message. At this stage, the client can send whichever messages it wants, and may receive messages from the server as a result. All these messages begin with a *message-type* byte, followed by any message-specific data.

The following descriptions of protocol messages use the basic types `CARD8`, `CARD16`, `CARD32`, `INT8`, `INT16`, `INT32`. These represent respectively 8, 16 and 32-bit unsigned integers and 8, 16 and 32-bit signed integers. All multiple byte integers (other than pixel values themselves) are in big endian order (most significant byte first).

5.1 Initial Handshaking Messages

5.1.1 ProtocolVersion

Handshaking begins by the server sending the client a *ProtocolVersion* message. This lets the client know which is the latest RFB protocol version number supported by the server. The client then replies with a similar message giving the version number of the protocol which should actually be used (which may be different to that quoted by the server).

It is intended that both clients and servers may provide some level of backwards compatibility by this mechanism. Servers in particular should attempt to provide backwards compatibility, and even forwards compatibility to some extent. For example if a client demands version 3.1 of the protocol, a 3.0 server can probably assume that by ignoring requests for encoding types it doesn't understand, everything will still work OK. This will probably not be the case for changes in the major version number.

The *ProtocolVersion* message consists of 12 bytes interpreted as a string of ASCII characters in the format `"RFB xxx.yyy\n"` where xxx and yyy are the major and minor version numbers, padded with zeros.

No. of bytes	Value
12	`"RFB 003.003\n"` (hex 52 46 42 20 30 30 33 2e 30 30 33 0a)

5.1.2 Authentication

Once the protocol version has been decided, the server then sends a word indicating the authentication scheme to be used on the connection:

No. of bytes	Type	[Value]	Description
4	CARD32		*authentication-scheme*:
		0	*connection failed*
		1	*no authentication*
		2	*VNC authentication*

This is followed by data specific to the *authentication-scheme*:

- ***connection failed*** - for some reason the connection failed (e.g. the server cannot support the desired protocol version). This is followed by a string describing the reason (where a string is specified as a length followed by that many ASCII characters):

No. of bytes	Type	[Value]	Description
4	CARD32		*reason-length*
reason-length	CARD8 array		*reason-string*

  The server closes the connection after sending the *reason-string*.

- ***no authentication*** - no authentication is needed. The protocol continues with the *ClientInitialisation* message.

- ***VNC authentication*** - VNC authentication is to be used. This is followed by a random 16-byte challenge:

No. of bytes	Type	[Value]	Description
16	CARD8		*challenge*

  The client encrypts the challenge with DES, using a password supplied by the user as the key, and sends the resulting 16-byte response:

No. of bytes	Type	[Value]	Description
16	CARD8		*response*

  The server sends a word to inform the client whether authentication was successful. If so, the protocol continues with the *ClientInitialisation* message; if not the server closes the connection:

No. of bytes	Type	[Value]	Description
4	CARD32		*status*:
		0	*OK*
		1	*failed*
		2	*too-many*

  If the server decides that *too-many* authentication failures have occurred, it should not allow immediate reconnection by the same client.

5.1.3 ClientInitialisation

Once the client and server are sure that they're happy to talk to one another, the client sends an initialisation message:

No. of bytes	Type [Value]	Description
1	CARD8	*shared-flag*

Shared-flag is non-zero (true) if the server should try to share the desktop by leaving other clients connected, zero (false) if it should give exclusive access to this client by disconnecting all other clients.

5.1.4 ServerInitialisation

After receiving the *ClientInitialisation* message, the server sends a *ServerInitialistion* message. This tells the client the width and height of the server's framebuffer, its pixel format and the name associated with the desktop:

No. of bytes	Type [Value]	Description
2	CARD16	*framebuffer-width*
2	CARD16	*framebuffer-height*
16	PIXEL_FORMAT	*server-pixel-format*
4	CARD32	*name-length*
name-length	CARD8 array	*name-string*

where PIXEL_FORMAT is

No. of bytes	Type [Value]	Description
1	CARD8	*bits-per-pixel*
1	CARD8	*depth*
1	CARD8	*big-endian-flag*
1	CARD8	*true-colour-flag*
2	CARD16	*red-max*
2	CARD16	*green-max*
2	CARD16	*blue-max*
1	CARD8	*red-shift*
1	CARD8	*green-shift*
1	CARD8	*blue-shift*
3		*padding*

Server-pixel-format specifies the server's natural pixel format. This pixel format will be used unless the client requests a different format using the *SetPixelFormat* message (section 5.2.1).

Bits-per-pixel is the number of bits used for each pixel value on the wire. This must be greater than or equal to the *depth* which is the number of useful bits in the pixel value. Currently *bits-per-pixel* must be 8, 16 or 32 — less than 8-bit pixels are not yet supported. *Big-endian-flag* is non-zero (true) if multi-byte pixels are interpreted as big endian. Of course this is meaningless for 8 bits-per-pixel.

If *true-colour-flag* is non-zero (true) then the last six items specify how to extract the red, green and blue intensities from the pixel value. *Red-max* is the maximum red value ($= 2^n - 1$ where n is the number of bits used for red). Note this value is always in big endian order. *Red-shift* is the number of shifts needed to get the red value in a pixel to the least significant bit. *Green-max*, *green-shift* and *blue-max*, *blue-shift* are similar for green and blue. For example, to find the red value (between 0 and *red-max*) from a given pixel, do the following:

- Swap the pixel value according to *big-endian-flag* (e.g. if *big-endian-flag* is zero (false) and host byte order is big endian, then swap).

- Shift right by *red-shift*.
- AND with *red-max* (in host byte order).

Currently there is little or no support for colour maps. Some preliminary work was done on this, but is incomplete. It was intended to be something like this:

> If *true-colour-flag* is zero (false) then the server uses pixel values which are not directly composed from the red, green and blue intensities, but which serve as indices into a colour map. Entries in the colour map can be set either by the client using the *FixColourMapEntries* message (section 5.2.2) or by the server using the *SetColourMapEntries* message (section 5.3.2).

The FixColourMapEntries message is not supported by any currently available servers, and colour maps are only supported at all by the X-based server. In fact, for proper colour map support the client probably needs to be able to specify particular pixel values which the server should not use. This may be added in future versions of the protocol, but don't hold your breath.

5.2 Client to server messages

5.2.1 SetPixelFormat

Sets the format in which pixel values should be sent in *FramebufferUpdate* messages. If the client does not send a *SetPixelFormat* message then the server sends pixel values in its natural format as specified in the *ServerInitialisation* message (section 5.1.4).

Currently there is little or no support for colour maps. Some preliminary work was done on this, but is incomplete. It was intended to be something like this:

> If *true-colour-flag* is zero (false) then this indicates that a "colour map" is to be used. The client can fix any of the entries in the colour map using the *FixColourMapEntries* message (section 5.2.2). Any entries not fixed by the client may be set dynamically as desired by the server using the *SetColourMapEntries* message (section 5.3.2). Immediately after the client has sent this message the colour map is empty, even if entries had previously been fixed by the client or set by the server.

No. of bytes	Type	[Value]	Description
1	CARD8	0	*message-type*
3			*padding*
16	PIXEL_FORMAT		*pixel-format*

where PIXEL_FORMAT is as described in section 5.1.4:

No. of bytes	Type	[Value]	Description
1	CARD8		*bits-per-pixel*
1	CARD8		*depth*
1	CARD8		*big-endian-flag*
1	CARD8		*true-colour-flag*
2	CARD16		*red-max*
2	CARD16		*green-max*
2	CARD16		*blue-max*
1	CARD8		*red-shift*
1	CARD8		*green-shift*
1	CARD8		*blue-shift*
3			*padding*

5.2.2 FixColourMapEntries

Currently there is little or no support for colour maps. Some preliminary work was done on this, but is incomplete. It was intended to be something like this:

> When the pixel format uses a "colour map", this message tells the server that the specified pixel values are mapped to the given RGB intensities. This means that the server may not subsequently specify RGB intensities for these pixel values using *SetColourMapEntries* messages (section 5.3.2).
>
> If the client has a fixed colour map it can simply send a *FixColourMapEntries* message describing its entire colour map and the server will then translate all pixel values as appropriate for the client's colour map. In this case the server cannot send any *SetColourMapEntries* messages.
>
> Note that despite the name, the client can, if it desires, send a *FixColourMapEntries* message at any time. This includes messages remapping the same pixel values to different RGB intensities. However, the only way the client can indicate that a pixel value is no longer fixed is by sending another *SetPixelFormat* message, which clears the entire colour map.

The FixColourMapEntries message is not supported by any currently available servers.

No. of bytes	Type	[Value]	Description
1	CARD8	1	*message-type*
1			*padding*
2	CARD16		*first-colour*
2	CARD16		*number-of-colours*

followed by *number-of-colours* repetitions of the following:

No. of bytes	Type	[Value]	Description
2	CARD16		*red*
2	CARD16		*green*
2	CARD16		*blue*

5.2.3 SetEncodings

Sets the encoding types in which pixel data can be sent by the server. The order of the encoding types given in this message is a hint by the client as to its preference (the first encoding specified being most preferred). The server may or may not choose to make use of this hint. Pixel data may always be sent in *raw* encoding even if not specified explicitly here.

No. of bytes	Type	[Value]	Description
1	CARD8	2	*message-type*
1			*padding*
2	CARD16		*number-of-encodings*

followed by *number-of-encodings* repetitions of the following:

No. of bytes	Type	[Value]	Description
4	CARD32		*encoding-type*
		0	*raw* encoding
		1	*copy rectangle* encoding
		2	*RRE* encoding
		4	*CoRRE* encoding
		5	*hextile* encoding

5.2.4 FramebufferUpdateRequest

Notifies the server that the client is interested in the area of the framebuffer specified by *x-position*, *y-position*, *width* and *height*. The server usually responds to a *FramebufferUpdateRequest* by sending a *FramebufferUpdate*. Note however that a single *FramebufferUpdate* may be sent in reply to several *FramebufferUpdateRequests*.

The server assumes that the client keeps a copy of all parts of the framebuffer in which it is interested. This means that normally the server only needs to send incremental updates to the client.

However, if for some reason the client has lost the contents of a particular area which it needs, then the client sends a *FramebufferUpdateRequest* with *incremental* set to zero (false). This requests that the server send the entire contents of the specified area as soon as possible. The area will not be updated using the *copy rectangle* encoding.

If the client has not lost any contents of the area in which it is interested, then it sends a *FramebufferUpdateRequest* with *incremental* set to non-zero (true). If and when there are changes to the specified area of the framebuffer, the server will send a *FramebufferUpdate*. Note that there may be an indefinite period between the *FramebufferUpdateRequest* and the *FramebufferUpdate*.

In the case of a fast client, the client may want to regulate the rate at which it sends incremental *FramebufferUpdateRequests* to avoid hogging the network.

No. of bytes	Type	[Value]	Description
1	CARD8	3	*message-type*
1	CARD8		*incremental*
2	CARD16		*x-position*
2	CARD16		*y-position*
2	CARD16		*width*
2	CARD16		*height*

Virtual Network Computing

AT&T

Home
Screenshots
Why is it free?
Getting started
Documentation
FAQ
Download
Keep in touch
Contributed
Want to help?
VNC people
Search
About us

Frequently Asked VNC Questions

This isn't intended to be an introduction to VNC - have a look at the Getting Started page first.

This list is being updated regularly, so, particularly if you are reading a local or mirrored copy, it's worth checking the original from time to time. The archives of the mailing list are also available, and they can be a good source of help - try searching them here.

Historical Note: With the release of 3.3.2 we have moved some of the old questions into the Old FAQ page because they should not apply to recent versions. As with most software, it's worth checking to make sure you've got the lastest version.

The FAQ is divided into the following sections:

- Problems getting started
  Installation errors, startup problems, basic communication difficulties, etc
- Problems running the programs
  Mouse problems, display issues etc.
- General questions
  How can I make VNC faster/more secure/run on my platform?
- Compiling the source
  Building VNC for yourself.

Problems Getting Started

Q. Where can I get VNC?

The latest versions of VNC and this documentation are always available from the AT&T Laboratories Cambridge web site at http://www.uk.research.att.com/vnc . In addition, various add-ons, extras, and ports to other platforms can be found on the contribs page.

Q. I thought this was something to do with ORL? What's the Olivetti/Oracle link here?

In January 1999, AT&T acquired ORL, the Olivetti Research Laboratory founded 12 years earlier, and recently jointly funded by Oracle, to create AT&T Laboratories Cambridge.

Q. I downloaded the .EXE files and they don't work!

hen it thinks it knows what type it is. W you fill in the download form you specify the type of compression you would like to use, and this does not include .EXE, so if the thing you downloaded has that extension, change it to whatever you requested (usually .tgz) and then unpack the files using an appropriate decoder.

Q. I unpacked the files on NT and ran SETUP.EXE but nothing happens.

Make sure you have NT service pack 3 installed. It appears that the InstallShield system we use won't work without it.

Q. WinVNC causes a 'Blue Screen of Death' on NT!

Again, make sure you have Service Pack 3 or later. If this happens it is usually due to bugs in Service Pack 1. Also note the Service Packs' warning that if you add or change components on your system you should reapply the service pack. If this doesn't fix it, check for updates to your network and graphics drivers. VNC makes very extensive use of both your video system and your network, and has a tendency to find any bugs in either of them! There may well be bugs in WinVNC, but we know of people running it without problems on hundreds of machines, so please check other parts of your system before assuming it's directly a WinVNC problem.

Q. How do I set up a Windows 95/98 machine so that I can dial into it directly and view it using VNC?

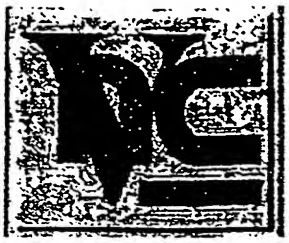
You need to install your modem from the control panel if you haven't already, and you need to set up the dialup networking server on your server machine. (This is included with Win98 and NT4. On Win95 it is in the Plus! pack, but you need to get an update to version 1.3 or later from Microsoft's site. At the time of writing it can be found here.) You can enable the dialup server from the 'Connections' menu of the dial-up networking window. If it isn't there, or if you've updated the dialup networking as mentioned above, you need to install it using the Windows Setup section of 'Add/Remove Programs' in the control panel. When it's running and you've dialled in, the server machine should have an IP address something like 192.168.55.1 as seen from the viewer - you can find this out by hovering over the VNC server icon, using 'netstat -r', or running winipcfg. You should then be able to connect to, for example, 192.168.55.1:0.

Q. I ran the vncserver program and got a 'No such file or directory' message.

Vncserver is a Perl script and so the first line has to point to the place where Perl is installed. On Linux this will generally be `/usr/bin/perl`, on other platfroms it's more likely to be `/usr/local/bin/perl` Edit the line to point to the right place and it should work.

Q. I ran the vncserver program and got a 'Number found where operator expected', 'prototype mismatch' or 'out of memory' message.

Note: This problem should be fixed in 3.3.2R3 and later. This is a Perl error, and happens on platforms where the Perl installation is not quite correct. We've heard of it particularly on S.U.S.E. linux and Redhat 5.2. The problem is



Virtual Network Computing

AT&T

VNC - the internal AT&T version

Some of the functionality of the distributed VNC system is limited when compared to the version we use within AT&T Laboratories Cambridge. This is not because we wish to deprive the rest of the world of a more sophisticated system, but because we want VNC to be easy to download and set up, and this extra functionality would make that process more complicated without offering major gains to most users.

So this information is given partly for interest's sake, and partly to explain some things you may find in the source code. If there is overwhelming demand for these features we may make them available, but this is not currently planned.

CORBA

Many of our systems are now linked together using CORBA. We didn't find a commercial implementation of CORBA which met our requirements, and so we wrote our own, omniORB, which we have also made freely available. The VNC servers have a CORBA interface which allows them to be controlled by remote systems. In particular, the interface exposes the name, address and other details of the server, allows control of the client connections, can perform remote shutdown, and initiate outgoing connections.

Outgoing connections

The simplest mode of operation for VNC is when a viewer initiates a connection to the server, but it can also operate in reverse. Some of our clients can be put into a 'listening' mode where they wait for connections from the server which are initiated and closed down using the CORBA interface.

External control

This means that desktops can be displayed and hidden under the control of other systems. In particular we use information from our Active Badge system to display a users' desktop on a nearby monitor when they click one of the buttons on the badge, and with our Ultrasound Location System we can make your desktop appear simply by walking up to a display!

```
 * Return or Enter      0xff0d
 * Escape               0xff1b
 * Insert               0xff63
 * Delete               0xffff
 * Home                 0xff50
 * End                  0xff57
 * Page Up              0xff55
 * Page Down            0xff56
 * Left                 0xff51
 * Up                   0xff52
 * Right                0xff53
 * Down                 0xff54
 * F1                   0xffbe
 * F2                   0xffbf
 * ...                  ...
 * F12                  0xffc9
 * Shift                0xffe1
 * Control              0xffe3
 * Meta                 0xffe7
 * Alt                  0xffe9
 */

typedef struct {
    CARD8 type;                 /* always rfbKeyEvent */
    CARD8 down;                 /* true if down (press), false if up */
    CARD16 pad;
    CARD32 key;                 /* key is specified as an X keysym */
} rfbKeyEventMsg;

#define sz_rfbKeyEventMsg 8


/*-----------------------------------------------------------------------------
 * PointerEvent - mouse/pen move and/or button press.
 */

typedef struct {
    CARD8 type;                 /* always rfbPointerEvent */
    CARD8 buttonMask;           /* bits 0-7 are buttons 1-8, 0=up, 1=down */
    CARD16 x;
    CARD16 y;
} rfbPointerEventMsg;

#define rfbButton1Mask 1
#define rfbButton2Mask 2
#define rfbButton3Mask 4

#define sz_rfbPointerEventMsg 6


/*-----------------------------------------------------------------------------
 * ClientCutText - the client has new text in its cut buffer.
 */

typedef struct {
    CARD8 type;                 /* always rfbClientCutText */
    CARD8 pad1;
    CARD16 pad2;
```

```
    CARD32 length;
    /* followed by char text[length] */
} rfbClientCutTextMsg;

#define sz_rfbClientCutTextMsg 8


/*-----------------------------------------------------------------------------
 * Union of all client->server messages.
 */

typedef union {
    CARD8 type;
    rfbSetPixelFormatMsg spf;
    rfbFixColourMapEntriesMsg fcme;
    rfbSetEncodingsMsg se;
    rfbFramebufferUpdateRequestMsg fur;
    rfbKeyEventMsg ke;
    rfbPointerEventMsg pe;
    rfbClientCutTextMsg cct;
} rfbClientToServerMsg;
```

```
/*-----------------------------------------------------------------------------
 * ServerCutText - the server has new text in its cut buffer.
 */

typedef struct {
    CARD8 type;                 /* always rfbServerCutText */
    CARD8 pad1;
    CARD16 pad2;
    CARD32 length;
    /* followed by char text[length] */
} rfbServerCutTextMsg;

#define sz_rfbServerCutTextMsg 8


/*-----------------------------------------------------------------------------
 * Union of all server->client messages.
 */

typedef union {
    CARD8 type;
    rfbFramebufferUpdateMsg fu;
    rfbSetColourMapEntriesMsg scme;
    rfbBellMsg b;
    rfbServerCutTextMsg sct;
} rfbServerToClientMsg;



/*****************************************************************************
 *
 * Message definitions (client -> server)
 *
 *****************************************************************************/


/*-----------------------------------------------------------------------------
 * SetPixelFormat - tell the RFB server the format in which the client wants
 * pixels sent.
 */

typedef struct {
    CARD8 type;                 /* always rfbSetPixelFormat */
    CARD8 pad1;
    CARD16 pad2;
    rfbPixelFormat format;
} rfbSetPixelFormatMsg;

#define sz_rfbSetPixelFormatMsg (sz_rfbPixelFormat + 4)


/*-----------------------------------------------------------------------------
 * FixColourMapEntries - when the pixel format uses a "colour map", fix
 * read-only colour map entries.
 *
 ********** NOT CURRENTLY SUPPORTED **********
```

```
 */

typedef struct {
    CARD8 type;                 /* always rfbFixColourMapEntries */
    CARD8 pad;
    CARD16 firstColour;
    CARD16 nColours;

    /* Followed by nColours * 3 * CARD16
       r1, g1, b1, r2, g2, b2, r3, g3, b3, ..., rn, bn, gn */

} rfbFixColourMapEntriesMsg;

#define sz_rfbFixColourMapEntriesMsg 6


/*-----------------------------------------------------------------------------
 * SetEncodings - tell the RFB server which encoding types we accept.  Put them
 * in order of preference, if we have any.  We may always receive raw
 * encoding, even if we don't specify it here.
 */

typedef struct {
    CARD8 type;                 /* always rfbSetEncodings */
    CARD8 pad;
    CARD16 nEncodings;
    /* followed by nEncodings * CARD32 encoding types */
} rfbSetEncodingsMsg;

#define sz_rfbSetEncodingsMsg 4


/*-----------------------------------------------------------------------------
 * FramebufferUpdateRequest - request for a framebuffer update.  If incremental
 * is true then the client just wants the changes since the last update.  If
 * false then it wants the whole of the specified rectangle.
 */

typedef struct {
    CARD8 type;                 /* always rfbFramebufferUpdateRequest */
    CARD8 incremental;
    CARD16 x;
    CARD16 y;
    CARD16 w;
    CARD16 h;
} rfbFramebufferUpdateRequestMsg;

#define sz_rfbFramebufferUpdateRequestMsg 10


/*-----------------------------------------------------------------------------
 * KeyEvent - key press or release
 *
 * Keys are specified using the "keysym" values defined by the X Window System.
 * For most ordinary keys, the keysym is the same as the corresponding ASCII
 * value.  Other common keys are:
 *
 * BackSpace            0xff08
 * Tab                  0xff09
```

```
/*-----------------------------------------------------------------------
 * RRE - Rise-and-Run-length Encoding.  We have an rfbRREHeader structure
 * giving the number of subrectangles following.  Finally the data follows in
 * the form [<bgpixel><subrect><subrect>...] where each <subrect> is
 * [<pixel><rfbRectangle>].
 */

typedef struct {
    CARD32 nSubrects;
} rfbRREHeader;

#define sz_rfbRREHeader 4


/*-----------------------------------------------------------------------
 * CoRRE - Compact RRE Encoding.  We have an rfbRREHeader structure giving
 * the number of subrectangles following.  Finally the data follows in the form
 * [<bgpixel><subrect><subrect>...] where each <subrect> is
 * [<pixel><rfbCoRRERectangle>].  This means that
 * the whole rectangle must be at most 255x255 pixels.
 */

typedef struct {
    CARD8 x;
    CARD8 y;
    CARD8 w;
    CARD8 h;
} rfbCoRRERectangle;

#define sz_rfbCoRRERectangle 4


/*-----------------------------------------------------------------------
 * Hextile Encoding.  The rectangle is divided up into "tiles" of 16x16 pixels,
 * starting at the top left going in left-to-right, top-to-bottom order.  If
 * the width of the rectangle is not an exact multiple of 16 then the width of
 * the last tile in each row will be correspondingly smaller.  Similarly if the
 * height is not an exact multiple of 16 then the height of each tile in the
 * final row will also be smaller.  Each tile begins with a "subencoding" type
 * byte, which is a mask made up of a number of bits.  If the Raw bit is set
 * then the other bits are irrelevant; w*h pixel values follow (where w and h
 * are the width and height of the tile).  Otherwise the tile is encoded in a
 * similar way to RRE, except that the position and size of each subrectangle
 * can be specified in just two bytes.  The other bits in the mask are as
 * follows:
 *
 * BackgroundSpecified - if set, a pixel value follows which specifies
 *    the background colour for this tile.  The first non-raw tile in a
 *    rectangle must have this bit set.  If this bit isn't set then the
 *    background is the same as the last tile.
 *
 * ForegroundSpecified - if set, a pixel value follows which specifies
 *    the foreground colour to be used for all subrectangles in this tile.
 *    If this bit is set then the SubrectsColoured bit must be zero.
 *
 * AnySubrects - if set, a single byte follows giving the number of
 *    subrectangles following.  If not set, there are no subrectangles (i.e.
 *    the whole tile is just solid background colour).
 *
```

```
 * SubrectsColoured - if set then each subrectangle is preceded by a pixel
 *    value giving the colour of that subrectangle.  If not set, all
 *    subrectangles are the same colour, the foreground colour;  if the
 *    ForegroundSpecified bit wasn't set then the foreground is the same as
 *    the last tile.
 *
 * The position and size of each subrectangle is specified in two bytes.  The
 * Pack macros below can be used to generate the two bytes from x, y, w, h,
 * and the Extract macros can be used to extract the x, y, w, h values from
 * the two bytes.
 */

#define rfbHextileRaw                   (1 << 0)
#define rfbHextileBackgroundSpecified   (1 << 1)
#define rfbHextileForegroundSpecified   (1 << 2)
#define rfbHextileAnySubrects           (1 << 3)
#define rfbHextileSubrectsColoured      (1 << 4)

#define rfbHextilePackXY(x,y) (((x) << 4) | (y))
#define rfbHextilePackWH(w,h) ((((w)-1) << 4) | ((h)-1))
#define rfbHextileExtractX(byte) ((byte) >> 4)
#define rfbHextileExtractY(byte) ((byte) & 0xf)
#define rfbHextileExtractW(byte) (((byte) >> 4) + 1)
#define rfbHextileExtractH(byte) (((byte) & 0xf) + 1)


/*-----------------------------------------------------------------------------
 * SetColourMapEntries - these messages are only sent if the pixel
 * format uses a "colour map" (i.e. trueColour false) and the client has not
 * fixed the entire colour map using FixColourMapEntries.  In addition they
 * will only start being sent after the client has sent its first
 * FramebufferUpdateRequest.  So if the client always tells the server to use
 * trueColour then it never needs to process this type of message.
 */

typedef struct {
    CARD8 type;                 /* always rfbSetColourMapEntries */
    CARD8 pad;
    CARD16 firstColour;
    CARD16 nColours;

    /* Followed by nColours * 3 * CARD16
       r1, g1, b1, r2, g2, b2, r3, g3, b3, ..., rn, bn, gn */

} rfbSetColourMapEntriesMsg;

#define sz_rfbSetColourMapEntriesMsg 6


/*-----------------------------------------------------------------------------
 * Bell - ring a bell on the client if it has one.
 */

typedef struct {
    CARD8 type;                 /* always rfbBell */
} rfbBellMsg;

#define sz_rfbBellMsg 1
```

```
/*
 * Following the server initialisation message it's up to the client to send
 * whichever protocol messages it wants.  Typically it will send a
 * SetPixelFormat message and a SetEncodings message, followed by a
 * FramebufferUpdateRequest.  From then on the server will send
 * FramebufferUpdate messages in response to the client's
 * FramebufferUpdateRequest messages.  The client should send
 * FramebufferUpdateRequest messages with incremental set to true when it has
 * finished processing one FramebufferUpdate and is ready to process another.
 * With a fast client, the rate at which FramebufferUpdateRequests are sent
 * should be regulated to avoid hogging the network.
 */


/*****************************************************************************
 *
 * Message types
 *
 *****************************************************************************/

/* server -> client */

#define rfbFramebufferUpdate 0
#define rfbSetColourMapEntries 1
#define rfbBell 2
#define rfbServerCutText 3


/* client -> server */

#define rfbSetPixelFormat 0
#define rfbFixColourMapEntries 1        /* not currently supported */
#define rfbSetEncodings 2
#define rfbFramebufferUpdateRequest 3
#define rfbKeyEvent 4
#define rfbPointerEvent 5
#define rfbClientCutText 6



/*****************************************************************************
 *
 * Encoding types
 *
 *****************************************************************************/

#define rfbEncodingRaw 0
#define rfbEncodingCopyRect 1
#define rfbEncodingRRE 2
#define rfbEncodingCoRRE 4
#define rfbEncodingHextile 5


/*****************************************************************************
```

```
 *
 * Server -> client message definitions
 *
 ************************************************************************/

/*-----------------------------------------------------------------------
 * FramebufferUpdate - a block of rectangles to be copied to the framebuffer.
 *
 * This message consists of a header giving the number of rectangles of pixel
 * data followed by the rectangles themselves.  The header is padded so that
 * together with the type byte it is an exact multiple of 4 bytes (to help
 * with alignment of 32-bit pixels):
 */

typedef struct {
    CARD8 type;                 /* always rfbFramebufferUpdate */
    CARD8 pad;
    CARD16 nRects;
    /* followed by nRects rectangles */
} rfbFramebufferUpdateMsg;

#define sz_rfbFramebufferUpdateMsg 4

/*
 * Each rectangle of pixel data consists of a header describing the position
 * and size of the rectangle and a type word describing the encoding of the
 * pixel data, followed finally by the pixel data.  Note that if the client has
 * not sent a SetEncodings message then it will only receive raw pixel data.
 * Also note again that this structure is a multiple of 4 bytes.
 */

typedef struct {
    rfbRectangle r;
    CARD32 encoding;    /* one of the encoding types rfbEncoding... */
} rfbFramebufferUpdateRectHeader;

#define sz_rfbFramebufferUpdateRectHeader (sz_rfbRectangle + 4)


/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 * Raw Encoding.  Pixels are sent in top-to-bottom scanline order,
 * left-to-right within a scanline with no padding in between.
 */


/*- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 * CopyRect Encoding.  The pixels are specified simply by the x and y position
 * of the source rectangle.
 */

typedef struct {
    CARD16 srcX;
    CARD16 srcY;
} rfbCopyRect;

#define sz_rfbCopyRect 4
```

```
#define sz_rfbPixelFormat 16


/*****************************************************************************
 *
 * Initial handshaking messages
 *
 *****************************************************************************/

/*-----------------------------------------------------------------------------
 * Protocol Version
 *
 * The server always sends 12 bytes to start which identifies the latest RFB
 * protocol version number which it supports.  These bytes are interpreted
 * as a string of 12 ASCII characters in the format "RFB xxx.yyy\n" where
 * xxx and yyy are the major and minor version numbers (for version 3.3
 * this is "RFB 003.003\n").
 *
 * The client then replies with a similar 12-byte message giving the version
 * number of the protocol which should actually be used (which may be different
 * to that quoted by the server).
 *
 * It is intended that both clients and servers may provide some level of
 * backwards compatibility by this mechanism.  Servers in particular should
 * attempt to provide backwards compatibility, and even forwards compatibility
 * to some extent.  For example if a client demands version 3.1 of the
 * protocol, a 3.0 server can probably assume that by ignoring requests for
 * encoding types it doesn't understand, everything will still work OK.  This
 * will probably not be the case for changes in the major version number.
 *
 * The format string below can be used in sprintf or sscanf to generate or
 * decode the version string respectively.
 */

#define rfbProtocolVersionFormat "RFB %03d.%03d\n"
#define rfbProtocolMajorVersion 3
#define rfbProtocolMinorVersion 3

typedef char rfbProtocolVersionMsg[13]; /* allow extra byte for null */

#define sz_rfbProtocolVersionMsg 12


/*-----------------------------------------------------------------------------
 * Authentication
 *
 * Once the protocol version has been decided, the server then sends a 32-bit
 * word indicating whether any authentication is needed on the connection.
 * The value of this word determines the authentication scheme in use.  For
 * version 3.0 of the protocol this may have one of the following values:
 */

#define rfbConnFailed 0
#define rfbNoAuth 1
#define rfbVncAuth 2

/*
 * rfbConnFailed:  For some reason the connection failed (e.g. the server
```

```
 *                          cannot support the desired protocol version).  This is
 *                          followed by a string describing the reason (where a
 *                          string is specified as a 32-bit length followed by that
 *                          many ASCII characters).
 *
 * rfbNoAuth:               No authentication is needed.
 *
 * rfbVncAuth:              The VNC authentication scheme is to be used.  A 16-byte
 *                          challenge follows, which the client encrypts as
 *                          appropriate using the password and sends the resulting
 *                          16-byte response.  If the response is correct, the
 *                          server sends the 32-bit word rfbVncAuthOK.  If a simple
 *                          failure happens, the server sends rfbVncAuthFailed and
 *                          closes the connection. If the server decides that too
 *                          many failures have occurred, it sends rfbVncAuthTooMany
 *                          and closes the connection.  In the latter case, the
 *                          server should not allow an immediate reconnection by
 *                          the client.
 */

#define rfbVncAuthOK 0
#define rfbVncAuthFailed 1
#define rfbVncAuthTooMany 2


/*-----------------------------------------------------------------------------
 * Client Initialisation Message
 *
 * Once the client and server are sure that they're happy to talk to one
 * another, the client sends an initialisation message.  At present this
 * message only consists of a boolean indicating whether the server should try
 * to share the desktop by leaving other clients connected, or give exclusive
 * access to this client by disconnecting all other clients.
 */

typedef struct {
    CARD8 shared;
} rfbClientInitMsg;

#define sz_rfbClientInitMsg 1


/*-----------------------------------------------------------------------------
 * Server Initialisation Message
 *
 * After the client initialisation message, the server sends one of its own.
 * This tells the client the width and height of the server's framebuffer,
 * its pixel format and the name associated with the desktop.
 */

typedef struct {
    CARD16 framebufferWidth;
    CARD16 framebufferHeight;
    rfbPixelFormat format;      /* the server's preferred pixel format */
    CARD32 nameLength;
    /* followed by char name[nameLength] */
} rfbServerInitMsg;

#define sz_rfbServerInitMsg (8 + sz_rfbPixelFormat)
```

```
/*
 *  Copyright (C) 1997, 1998 Olivetti & Oracle Research Laboratory
 *
 *  This is free software; you can redistribute it and/or modify
 *  it under the terms of the GNU General Public License as published by
 *  the Free Software Foundation; either version 2 of the License, or
 *  (at your option) any later version.
 *
 *  This software is distributed in the hope that it will be useful,
 *  but WITHOUT ANY WARRANTY; without even the implied warranty of
 *  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *  GNU General Public License for more details.
 *
 *  You should have received a copy of the GNU General Public License
 *  along with this software; if not, write to the Free Software
 *  Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307,
 *  USA.
 */

/*
 * rfbproto.h - header file for the RFB protocol version 3.3
 *
 * Uses types CARD<n> for an n-bit unsigned integer, INT<n> for an n-bit signed
 * integer (for n = 8, 16 and 32).
 *
 * All multiple byte integers are in big endian (network) order (most
 * significant byte first).  Unless noted otherwise there is no special
 * alignment of protocol structures.
 *
 *
 * Once the initial handshaking is done, all messages start with a type byte,
 * (usually) followed by message-specific data.  The order of definitions in
 * this file is as follows:
 *
 *   (1) Structures used in several types of message.
 *   (2) Structures used in the initial handshaking.
 *   (3) Message types.
 *   (4) Encoding types.
 *   (5) For each message type, the form of the data following the type byte.
 *       Sometimes this is defined by a single structure but the more complex
 *       messages have to be explained by comments.
 */


/*****************************************************************************
 *
 * Structures used in several messages
 *
 *****************************************************************************/

/*-----------------------------------------------------------------------------
 * Structure used to specify a rectangle.  This structure is a multiple of 4
 * bytes so that it can be interspersed with 32-bit pixel data without
 * affecting alignment.
 */

typedef struct {
    CARD16 x;
    CARD16 y;
```

```
    CARD16 w;
    CARD16 h;
} rfbRectangle;

#define sz_rfbRectangle 8


/*-----------------------------------------------------------------------------
 * Structure used to specify pixel format.
 */

typedef struct {

    CARD8 bitsPerPixel;          /* 8,16,32 only */

    CARD8 depth;                 /* 8 to 32 */

    CARD8 bigEndian;             /* True if multi-byte pixels are interpreted
                                    as big endian, or if single-bit-per-pixel
                                    has most significant bit of the byte
                                    corresponding to first (leftmost) pixel. Of
                                    course this is meaningless for 8 bits/pix */

    CARD8 trueColour;            /* If false then we need a "colour map" to
                                    convert pixels to RGB.  If true, xxxMax and
                                    xxxShift specify bits used for red, green
                                    and blue */

    /* the following fields are only meaningful if trueColour is true */

    CARD16 redMax;               /* maximum red value (= 2^n - 1 where n is the
                                    number of bits used for red). Note this
                                    value is always in big endian order. */

    CARD16 greenMax;             /* similar for green */

    CARD16 blueMax;              /* and blue */

    CARD8 redShift;              /* number of shifts needed to get the red
                                    value in a pixel to the least significant
                                    bit. To find the red value from a given
                                    pixel, do the following:
                                    1) Swap pixel value according to bigEndian
                                       (e.g. if bigEndian is false and host byte
                                       order is big endian, then swap).
                                    2) Shift right by redShift.
                                    3) AND with redMax (in host byte order).
                                    4) You now have the red value between 0 and
                                       redMax. */

    CARD8 greenShift;            /* similar for green */

    CARD8 blueShift;             /* and blue */

    CARD8 pad1;
    CARD16 pad2;

} rfbPixelFormat;
```

5.3.3 Bell

Ring a bell on the client if it has one.

No. of bytes	Type	[Value]	Description
1	CARD8	2	*message-type*

5.3.4 ServerCutText

The server has new ASCII text in its cut buffer. End of lines are represented by the linefeed / newline character (ASCII value 10) alone. No carriage-return (ASCII value 13) is needed.

No. of bytes	Type	[Value]	Description
1	CARD8	3	*message-type*
3			*padding*
4	CARD32		*length*
length	CARD8 array		*text*

No. of bytes	Type [Value]	Description
n	CARD<8n>	*background-pixel-value*

where 8*n* is the number of *bits-per-pixel* as agreed by the client and server – either in the *ServerInitialisation* message (section 5.1.4) or a *SetPixelFormat* message (section 5.2.1). The first non-raw tile in a rectangle must have this bit set. If this bit isn't set then the background is the same as the last tile.

ForegroundSpecified - if set, a pixel value follows which specifies the foreground colour to be used for all subrectangles in this tile:

No. of bytes	Type [Value]	Description
n	CARD<8n>	*foreground-pixel-value*

If this bit is set then the SubrectsColoured bit must be zero.

AnySubrects - if set, a single byte follows giving the number of subrectangles following:

No. of bytes	Type [Value]	Description
1	CARD8	*number-of-subrectangles*

If not set, there are no subrectangles (i.e. the whole tile is just solid background colour).

SubrectsColoured - if set then each subrectangle is preceded by a pixel value giving the colour of that subrectangle, so a subrectangle is:

No. of bytes	Type [Value]	Description
n	CARD<8n>	*subrect-pixel-value*
1	CARD8	*x-and-y-position*
1	CARD8	*width-and-height*

If not set, all subrectangles are the same colour, the foreground colour; if the ForegroundSpecified bit wasn't set then the foreground is the same as the last tile. A subrectangle is:

No. of bytes	Type [Value]	Description
1	CARD8	*x-and-y-position*
1	CARD8	*width-and-height*

The position and size of each subrectangle is specified in two bytes, *x-and-y-position* and *width-and-height*. The most-significant four bits of *x-and-y-position* specify the X position, the least-significant specify the Y position. The most-significant four bits of *width-and-height* specify the width minus one, the least-significant specify the height minus one.

5.3.2 SetColourMapEntries

Currently there is little or no support for colour maps. Some preliminary work was done on this, but is incomplete. It was intended to be something like this:

> When the pixel format uses a "colour map", this message tells the client that the specified pixel values should be mapped to the given RGB intensities. The server may only specify pixel values for which the client has not already set the RGB intensities using *FixColourMapEntries* (section 5.2.2).

Currently, colour maps are only supported at all by the X-based server.

No. of bytes	Type	[Value]	Description
1	CARD8	1	*message-type*
1			*padding*
2	CARD16		*first-colour*
2	CARD16		*number-of-colours*

followed by *number-of-colours* repetitions of the following:

No. of bytes	Type	[Value]	Description
2	CARD16		*red*
2	CARD16		*green*
2	CARD16		*blue*

5.3.1 FramebufferUpdate

A framebuffer update consists of a sequence of rectangles of pixel data which the client should put into its framebuffer. It is sent in response to a *FramebufferUpdateRequest* from the client. Note that there may be an indefinite period between the *FramebufferUpdateRequest* and the *FramebufferUpdate*.

No. of bytes	Type	[Value]	Description
1	CARD8	0	*message-type*
1			*padding*
2	CARD16		*number-of-rectangles*

This is followed by *number-of-rectangles* rectangles of pixel data. Each rectangle consists of:

No. of bytes	Type	[Value]	Description
2	CARD16		*x-position*
2	CARD16		*y-position*
2	CARD16		*width*
2	CARD16		*height*
4	CARD32		*encoding-type*:
		0	***raw*** encoding
		1	***copy rectangle*** encoding
		2	***RRE*** encoding
		4	***CoRRE*** encoding
		5	***hextile*** encoding

followed by the pixel data in the specified encoding.

- For ***raw*** encoding the data consists of *width* × *height* pixel values. The values simply represent each pixel in left-to-right scanline order.
- For ***copy rectangle*** encoding the data consists of:

No. of bytes	Type	[Value]	Description
2	CARD16		*src-x-position*
2	CARD16		*src-y-position*

- For ***RRE*** encoding the data begins with the header:

No. of bytes	Type	[Value]	Description
4	CARD32		*number-of-subrectangles*
n	CARD<8n>		*background-pixel-value*

where 8*n* is the number of *bits-per-pixel* as agreed by the client and server – either in the *ServerInitialisation* message (section 5.1.4) or a *SetPixelFormat* message (section 5.2.1). This is followed by *number-of-subrectangles* instances of the following structure:

No. of bytes	Type [Value]	Description
n	CARD<8n>	*subrect-pixel-value*
2	CARD16	*x-position*
2	CARD16	*y-position*
2	CARD16	*width*
2	CARD16	*height*

- For ***CoRRE*** encoding the data begins with the header:

No. of bytes	Type [Value]	Description
4	CARD32	*number-of-subrectangles*
n	CARD<8n>	*background-pixel-value*

where 8*n* is the number of *bits-per-pixel* as agreed by the client and server – either in the *ServerInitialisation* message (section 5.1.4) or a *SetPixelFormat* message (section 5.2.1). This is followed by *number-of-subrectangles* instances of the following structure:

No. of bytes	Type [Value]	Description
n	CARD<8n>	*subrect-pixel-value*
1	CARD8	*x-position*
1	CARD8	*y-position*
1	CARD8	*width*
1	CARD8	*height*

- For ***hextile*** encoding the rectangle is divided up into *tiles* of 16x16 pixels, starting at the top left going in left-to-right, top-to-bottom order. If the width of the rectangle is not an exact multiple of 16 then the width of the last tile in each row will be correspondingly smaller. Similarly if the height is not an exact multiple of 16 then the height of each tile in the final row will also be smaller.

  Each tile begins with a *subencoding* type byte, which is a mask made up of a number of bits:

No. of bytes	Type	[Value]	Description
1	CARD8		*subencoding-mask*:
		1	*Raw*
		2	*BackgroundSpecified*
		4	*ForegroundSpecified*
		8	*AnySubrects*
		16	*SubrectsColoured*

If the *Raw* bit is set then the other bits are irrelevant; *width* × *height* pixel values follow (where *width* and *height* are the width and height of the tile). Otherwise the other bits in the mask are as follows:

BackgroundSpecified - if set, a pixel value follows which specifies the background colour for this tile:

5.2.7 ClientCutText

The client has new ASCII text in its cut buffer. End of lines are represented by the linefeed / newline character (ASCII value 10) alone. No carriage-return (ASCII value 13) is needed.

No. of bytes	Type	[Value]	Description
1	CARD8	6	*message-type*
3			*padding*
4	CARD32		*length*
length	CARD8 array		*text*

5.3 Server to client messages

5.2.5 KeyEvent

A key press or release. *Down-flag* is non-zero (true) if the key is now pressed, zero (false) if it is now released. The *key* itself is specified using the "keysym" values defined by the X Window System. For full details, see The Xlib Reference Manual, published by O'Reilly & Associates, or see the header file `<X11/keysymdef.h>` from any X Window System installation.

No. of bytes	Type	[Value]	Description
1	CARD8	4	*message-type*
1	CARD8		*down-flag*
2			*padding*
4	CARD32		*key*

For most ordinary keys, the "keysym" is the same as the corresponding ASCII value. Other common keys are:

Key name	Keysym value
BackSpace	0xff08
Tab	0xff09
Return or Enter	0xff0d
Escape	0xff1b
Insert	0xff63
Delete	0xffff
Home	0xff50
End	0xff57
Page Up	0xff55
Page Down	0xff56
Left	0xff51
Up	0xff52
Right	0xff53
Down	0xff54

Key name	Keysym value
F1	0xffbe
F2	0xffbf
F3	0xffc0
F4	0xffc1
...	...
F12	0xffc9
Shift (left)	0xffe1
Shift (right)	0xffe2
Control (left)	0xffe3
Control (right)	0xffe4
Meta (left)	0xffe7
Meta (right)	0xffe8
Alt (left)	0xffe9
Alt (right)	0xffea

5.2.6 PointerEvent

Indicates either pointer movement or a pointer button press or release. The pointer is now at (*x-position*, *y-position*), and the current state of buttons 1 to 8 are represented by bits 0 to 7 of *button-mask* respectively, 0 meaning up, 1 meaning down (pressed).

No. of bytes	Type	[Value]	Description
1	CARD8	5	*message-type*
1	CARD8		*button-mask*
2	CARD16		*x-position*
2	CARD16		*y-position*

Q. I connected to my Unix server and I just see a grey desktop with a cursor. Where's my normal X environment?

After the vncserver script has started the Xvnc server, it then runs your ~/.vnc/xstartup script. By default this will try to start the twm window manager, but if twm isn't on your path, or if you prefer something else, you can edit xstartup. The log file may also give you clues about what is happening.

Q. The log file is showing an error message from xrdb / Xlib.

By default, the first thing your xstartup script does is to run xrdb to load your resources. So if the Xvnc server has not started for any reason, the xrdb is often the first thing to notice it and print an error. (Though if you're getting a 'command not found' message, then xrdb is probably not on your path - you need to find where it is on your system and add it.)
If you get something like 'connection refused' or 'Can't connect: errno = 111', the Xvnc server probably isn't there. So you should check whether the Xvnc process is actually running, and whether there is anything earlier in the log file indicating why it might have died. By far the most common reason for the server not starting is that it can't find the 'fixed' font (see above). Other possibilities are that the server has quietly crashed, or that it is taking a very long time to start up. The vncserver script has a 3-second delay before running xstartup, but in extreme cases this may not be enough. Lastly, the DISPLAY variable used by vncserver is based on the results given by the 'uname - n' command. If your applications cannot resolve this to the right IP address, perhaps because of funny settings in /etc/hosts, then they won't be able to connect.
If you get a 'Client is not authorized to connect to Server' or similar message, there's something wrong with the X authority setup - perhaps xauth is not on your path? You could try using xhost to bypass this temporarily, but we wouldn't recomend this as a long-term solution. There should be some indication in the log file if xauth has failed.

Q. The Java client doesn't work in my browser.

Several Java implementations have bugs which upset the VNC applet. We recommend Internet Explorer 4 - we've had good results with this on both Windows machines and Macs. Netscape Navigator 3 has a bug which can cause problems; if you get the 'Method setClip not found' error, try pressing Reload.

Q. My viewer failed to connect to my server!

VNC relies on a correctly-configured and operational TCP/IP network, so please make 100% sure that your TCP/IP setup is right before you start asking questions on the mailing list. Here are some things you should check before assuming it's a VNC problem; consult your local expert if you don't know how to check them:

- Can you ping the server machine from the client?
- Is the VNC server definitely running on the server machine?
- The server listens on port 5900+displaynumber. Can you telnet to this port from the client machine?
- Have you specified the address correctly to the viewer? Did you remember the display number?
- Is the server name known to the DNS? Try using an explicit IP address

```
eval 'require "sys/socket.ph"';
```

Your first option is to change this to:

```
($] >= 5.0) ? eval 'use Socket' : eval 'require
"sys/socket.ph"';
```

If this doesn't work, particularly on S.u.S.E., you could try:

```
eval 'require "linux/socket.ph"';
```

or check your Perl installation to see if socket.ph is to be found in another directory.

Remember that vncserver is only a convenient wrapper around Xvnc. If it causes you problems, feel free to discard it and run Xvnc directly.

Q. I started the X server using vncserver, but it dies with a message "Could not open default font 'fixed'. "

The 'fixed' font is needed for the server to start - if it can't find it, you need to specify the correct font path for your machine in the 'vncserver' script. If you're not sure what the path should be, type 'xset q' from within a normal X session. One of the things reported is the font path used by your current X server, which is generally the right thing to use for Xvnc. On some platforms you may need to use a colon as a separator in the font path instead of a comma. If you're on a recent version of Linux but still using VNC version 3.3.1, you may have compressed fonts which VNC doesn't understand. Either upgrade or see the Old FAQ.

The VNC server can also get upset if you have directories on your font path which don't actually exist on your system. Make sure you remove those. Also note that the Xvnc server, by default, acts as if it has a resolution of 100dpi. Some RedHat installations, for example, only install 75dpi fonts, so you may need to install the missing font RPMs from your distribution or use the -dpi option to Xvnc.

Q. I get errors like "failed to bind listener" and "Failed to establish all listening sockets" in the log file.

This is probably due to the permissions on /tmp/.X11-unix. You may well see this if you update to Solaris 2.7, for example. See the section below entitled "Why can I only run vncserver/Xvnc as root?".

Q. Vncserver seems to be dying quietly without putting any messages in the log file.

Check that the Xvnc process really has died. If so, then check that your VNC font path (set by uncommenting lines in the vncserver script) only includes directories which actually exist. The XFree86 code in Xvnc seems to have a problem which causes the server to die quietly if non-existent directories are searched. You could also modify the vncserver script to check for and remove non-existent directories. See Bruce A. Mah's patch for an example.

Q. My Solaris/HP-UX Xvnc dies with a core dump!

There is a bug in XFree86 (on which Xvnc is based) which makes Solaris servers very unreliable if they have a pixel depth of 16. Use the -depth option to start your server with a depth of 8 or 24 and you should be fine. We've heard that this can be a problem with HP-UX as well. Darrell Kindred submitted a patch which speeds up operation on Alpha machines, and also

- Do you have any firewalls or proxies in the way that could be blocking acess?
- If using the java client, did you remember to specify the correct port as part of the URL? (eg. http://snoopy:5800)?
- Can you try running either the server, or the client, or both, on different machines on your network to find whether the problem is at one end or the other?
- Can you try running the software on a different architecture? eg., if you are having problems viewing a PC from another PC, can you try connecting from a Unix machine?

Problems running the programs

Q. My X VNC server is working, but I don't see my normal environment. How can I change the Window manager etc?

The window manager is started by the ~/.vnc/xstartup script. We use twm, as this is available on almost all Unix platforms. Edit the script if you'd rather replace it with something else. On many platforms you can, as an alternative, just make xstartup a link to whatever script normally starts your X environment. If you want to be more sophisticated, you can specify the -name option to vncserver, and then take different actions in the xstartup script based on the name given. For example:

```
case "$VNCDESKTOP" in

kde)

        startkde &

        ;;

*)

        xterm -geometry 40x10+40+40 -ls -title "$VNCDESKTOP D

        twm &

        ;;

esac
```

Q. Why can I only run vncserver/Xvnc as root?

The most likely reason for this is that Xvnc can't create the unix domain socket (the path for this unix domain socket is usually /tmp/.X11-unix/Xn). Try making sure that users can write to this directory by making it world-writable, i.e. "chmod a+w /tmp/.X11-unix". Note that to avoid a security loophole the "sticky bit" should also be set on the directory by doing "chmod o+t /tmp/.X11-unix". If you don't do this then someone else logged in to the same machine may be able to intercept the X protocol and thereby access your desktop and snoop on it, etc. An alternative is to set the Xvnc binary to have the same permissions as your normal X server, but this may be more of a security risk.

Q. Can I remote the normal X display of my workstation (display :0) in the same way as the Windows server does?

perated this way, but we have no plans to do so. It shouldn't be too difficult, if you have the full source code for your X server. Xvnc is basically XFree86 with the hardware-dependent bits taken out and VNC put in their place, so it doesn't have drivers for any graphics cards. You could have a look at the Xvnc source code to see which bits have been added to the standard XFree86, and make the equivalent changes on your X server.
We tend to run all our X sessions as VNC sessions and only use the local X server to run the viewer. It's very fast when on the same machine as the server! If you feel that it's overkill to run two X servers on the same machine, you might consider Ganesh Varadarajan's svgalib-based viewer, available from the contribs page.

Q. What X Visual does Xvnc use?

By default, vncserver will start Xvnc with the same depth as the current X display, if there is one, or 8 bits deep if there isn't. We've tried to steer clear c colour maps as much as possible and normally use "true colour", even when there are only 8 bits per pixel.
Unfortunately some X applications don't cope too well with an 8 bit TrueColor visual. You can make Xvnc use the more normal PseudoColor visual by giving a "-cc 3" option to vncserver.

Q. Can I cut and paste between the viewer and the server?

VNC supports copying and pasting of ASCII text in both directions, provided the viewer and server allow it. When the clipboard changes on the machine running the viewer, the changes are copied to the server and vice versa. Some notable exceptions:

- X has more than one method of using the clipboard and different applications do it different ways. Emacs and xterm should just work. If you find that your X application doesn't work via VNC, you can generally use the `xcutsel` program to copy the clipboard between the different X methods. VNC uses Cut_Buffer0, so if you select text in Unix Netscape, for example, you may need to click 'Copy PRIMARY to 0' before it is accessible at the other end of the VNC link.
- Java applets running in the browser cannot access the clipboard of the machine on which they are running, so the Java viewer has a clipboard button. This pops up a window displaying the contents of the remote clipboard, which should allow you to manipulate it locally.

The code to do copying from the X server to the viewer didn't make it in the original version 3.3.2. It's back in 3.3.2R2 and later, so upgrade if you haven't already!

Q. There's a memory leak in Xvnc!

This occurs in version 3.3.2r2 and earlier, but is fixed in 3.3.2r3 and later versions. If you're using an older distribution you can find a patch for it here

Q. Can I run the Windows server before anybody has logged in?

You can now take advantage of a recent WinVNC and read the section on running WinVNC as a service in the documentation.

Q. The keyboard doesn't work / keys do strange things!

There is one common problem which can cause this. If a modifier key, such as Shift, Ctrl or Alt, is pressed, and the viewer window then loses focus or dies, the 'key release' message never gets to the viewer and hence never gets to the remote server. The remote machine will then think that M is Ctrl-M etc. We have done various things to reduce the chance of this happening; the viewers release various modifiers automatically when they lose focus, for example, but it can still occur and can be confusing when it does. The solution is easy: simply press and release the modifier key which is stuck. If you don't know which it is, then try them one at a time.

Q. Most of my Windows apps work fine remotely, but this one doesn't update its window...

WinVNC can use a variety of hints to guess when a particular area of the screen has changed. The most useful is the occurrence of a WM_PAINT message in an application. Not all applications use these messages, though; the Windows clock is a good example.
In the registry WinVNC has a list of application names, and which things to use as update clues. A few standard applications are set up by the 'Install default registry settings' in the WinVNC section of the Start Menu; it tells WinVNC to look for timer messages from the clock, for example. Make sure you install these if you haven't already. Any other apps you run will appear in the registry, so you can easily tweak the settings. For more info, see the WinVNC documentation.

Q. WinVNC is putting a huge load on my PC!

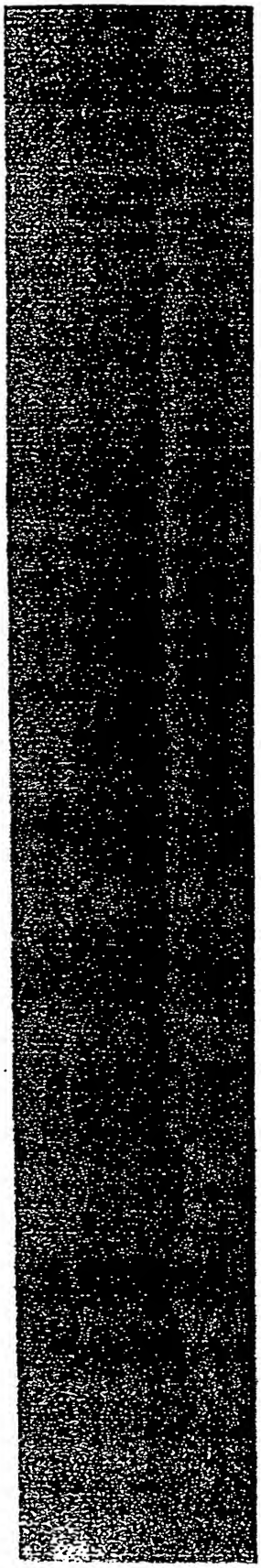
Firstly, WinVNC should have negligible impact when nobody is connected to it. It does practically nothing in this state. If you find that it is running at something close to 100% CPU when there is a remote connection ,check the Update Handling in the Properties dialog box. The default settings should be fine in most circumstances. If you have either (a) Ticked 'Poll Full Screen' or (b) Ticked 'Poll Foreground Window' or 'Poll Window Under Cursor' without having 'Poll Console Windows Only' ticked, then the load will be much higher. See also the section below on 'How do I make VNC go faster?'.

Q. WinVNC dies, or causes other applications to die, after a short time when a viewer is connected.

Some screen savers, particularly on Windows 95, do not interoperate well with WinVNC. Try disabling all screen savers on the machine running the server and see if that fixes it.

Q. My remote Windows display is appearing in a very garbled form.

There can be problems with WinVNC being unable to detect changes to the screen on certain applications, as mentioned in the previous question, but there should not generally be serious screen corruption. Make sure you have the latest service packs installed for your OS and experiment with different video drivers. WinVNC relies on various aspects of the video card and driver; in particular its ability to BitBlt correctly, and several users have solved their problems by updating the driver.



Q. Why can't I unlock my NT workstation remotely? Why can't I stop the screensaver remotely? Why doesn't Ctrl-Alt-Del work?

Make sure you are running a recent version of VNC, **and that you are running it as a service**. From some platforms you will not be able to type Ctrl-Alt-Del directly, because it will be caught by the local machine. The Windows viewer, for example, has an option on its menu to send a Ctrl-Alt-Del to the remote host. In some situations, you will find that something like Ctrl-Alt-Backspace or Ctrl-Alt-<Numeric keypad Del> may work instead. Screensavers sometimes use a different resolution and so can disconnect you when they stop or start - see the next question.

Q. When I connect using VNC and then log into my Windows machine, I get disconnected and have to reconnect!

Sometimes logging in will involve a change in screen resolution, if the user's display settings are different from the defaults. If this happens, the server will disconnect you and you will need to reconnect to get the new screen size. Just occasionally on NT, the mode seems to change resolution temporarily as you log in, and if WinVNC happens to see this you can also be disconnected, even if the final resolution changes.
If the user has set a different display number in their personal WinVNC properties dialog you will also be disconnected.

Q. I have troubles sending Ctrl-Alt-Del to a Windows server.

- Ctrl-Alt-Del will only be recognised by a Windows NT VNC server, and only when VNC is running as a service. On Win95/98, it suspends all processes, including the VNC server, so it wouldn't be much use!
- The Windows and Java viewers have menu options to send Ctrl-Alt-Del, so they don't get interpreted locally.
- If you are using the Mac or X-based viewer, you should just be able to type the keys, but some platforms seem to catch the keystrokes. If this is the case for you, I'm afraid you'll have to work out how to stop it! It might be worth trying slight variations (eg. the right Ctl key, or a different Del/Delete key, if you have one).

Q. The dead keys don't work on my keyboard

They will soon! It's quite a challenge working out how to do international keyboard support across different platforms. We're working on dead keys.

Q. When I start a DOS window, it doesn't display at the remote end.

You might also notice the pointer leaving a trail of arrows behind it. This happens when your DOS sessions are full-screen. WinVNC cannot read the display when this happens, but keystrokes should still get through. Press Alt-Enter to switch the DOS box to windowed mode.

Q. I can't type into a DOS window or any DOS apps

This derives from the fact that Windows 95 uses BIOS calls and not Windows messages to get keystrokes within the command prompt. It should be fixed in version 3.3.1 R19 and later of the WinVNC server, so if you're using an earlier

WinVNC server?

If your ISP allocates you a dynamic IP address when you dial in, you will need to give that to the person trying to connect. On Windows95 machines, after connecting, you can use the `winipcfg` program (type it into the Start/Run... box). On NT machines, type `ipconfig` at a command prompt. On linux machines, try `hostname -i`. A handy new feature on recent versions of WinVNC causes the IP addresses of the local machine to be displayed when the mouse hovers over the WinVNC icon, (if they can be determined at that time).
Of course, if your phone line is now in use you may need to find some other way of getting this information to the remote person. I suggest a chat system like AOL Instant Messenger. There are also programs out there which will automatically create a web page with your current IP address, which the other person could then read.

Q. Can I get rid of the taskbar icon created by WinVNC?

No. Not without changing and recompiling the source code. We feel that there would be few legitimate uses of VNC where you would need to conceal its operation. However, there might be occasions where you would not want the user of the server machine to be able to adjust the WinVNC settings, password, etc, so versions later than 3.3.2R3 will include a 'restricted' mode where the icon is still visible, but the menu options normally available from it will be disabled.
In some situations, particularly on Win95, you may find that the icon is not displayed. This is not a feature, it's a bug which we plan to iron out!

Q. Can I set up WinVNC to use my Windows NT password for authentication?

Not at present. Partly because there are many problems with the NT security model, but chiefly because we want to keep VNC as cross-platform as possible. In the future we may try and make the code and the protocol more modular so it will be easier to add your own favourite authentication.

Q. Can I make the Macintosh server start automatically when the machine boots up?

Yes, create an alias to VNCServer and put it into the Startup items folder (in your system folder).

Q. When I try to set a new password for my Mac server, I hit CHANGE on the web page, but nothing happens. It is trying to bring up the settings.html page, but fails.

The http portion of the current server is unstable, sorry. But you can set your settings with AppleScript.

Open the AppleScript editor and enter the following

```
tell application "VNCServer"
set password to "whatever"
end tell
```

By default, mouse button up events are used to signal to WinVNC that the current window may have changed and should be re-scanned. This can introduce a delay which results in a double-click being interpreted by the system as a pair of single clicks. Variations in network delays can also cause the clicks to be too widely separated when they appear at the server. Some suggestions:

- In many situations, the action performed by a double-click is also available from a right-button menu.
- You may find that clicking three times, rather than two, will help.
- You can change the double-click speed using the control panel on the server so that more widely-spaced clicks are still interpreted as doubles.
- You can disable the left-click hook by editing the registry, as described in the WinVNC documentation, although this may mean that WinVNC misses some screen updates from that application.

Q. I'm installing WinVNC as a service on lots of machines, and I get prompted for the password on each one. Can I do this non-interactively?

Not directly, but you can by editing the registry before installing. On a machine which already has WinVNC installed, copy all the registry settings under

```
HKEY_USERS/.Default/Software/ORL/WinVNC3
```

into the same place in the registry of the remote machine, either by hand or using your favourite tool.

When WinVNC runs, it will see the password in that section of the registry and will not prompt you to type one in.

Another hint that might be useful was sent in by Johannes Norinder. If you run the SETUP program with a -r option, it will create a setup.iss file, probably in your %systemroot% directory. Following installations on similar computer can be started with "setup -s -h1[path to your setup.iss]" and you won't have to answer the questions again!

Q. I only have a two-button Windows mouse, and I really need three buttons for X...

Get version 3.3.2R10 or later of the Windows viewer. This allows you to emulate the middle button by pressing both buttons together.

Q. I have a three-button Windows mouse, but the middle button doesn't work.

This is almost certainly a problem with your mouse driver. The Windows VNC viewer recognises standard WM_MBUTTON... messages and should work with any driver that generates them. Some drivers, knowing that Windows seldom normally uses the middle button, either don't recognise it, or map it to something else like a double-click. Try telling Windows that you have a different type of mouse, (we've had reports that the Logitech PS/2 Port Mouse is a good one to try) or use the 3-button emulation mode mentioned above. I think some PS/2 mouse port hardware may also not recognise the middle button.

- If you are connecting to WinVNC, don't change the default settings in the Properties box unless you need to. See 'WinVNC is putting a huge load on my PC' above.
- All the standard clients can do local rectangle copying, and this generally means that dragging a full window on X is much faster than dragging its outline. If your window manager allows this, it may speed things up. Roman Mitnitski also reports that when he increased the mouse dragging threshold (-t option of the server) the performance was improved.
- Generally, with WinVNC, use 16-bit colour (65536 colours) on the server if you can. 16-bit is almost always the best depth to use, because:
  1. 256-colour screens have to be palette-converted before they can be transmitted to truecolour clients. Only if the client is 256-colour palette- based will you see any performance increase. Even if the client is 256 colour truecolour, it'll have to convert via a 32-bit truecolour palette!
  2. 24-bit screens have to be specially munged via 32-bit since VNC's internal colour-handling routines don't work with 24-bit directly.
  3. 24 and 32-bit screens have to have each pixel looked up in three tables to get the converted value.
  4. Graphics cards claiming to do 24-bit often actually do 32-bit with munging - this in many cases makes 24-bit slower just for general use than 32-bit!
  5. Finally, 16-bit involves no palette processing and a single lookup in a cached src_format to dest_format table to convert the pixels.

On slow links, you may also want to use some software to compress the data between the two ends. SSH is good for this. See the section on security below.

Q. Will VNC work through a firewall?

It depends on your firewall, and whether you want to access a server inside your firewall from elsewhere, or a server outside your firewall from inside. Generally firewalls are designed to prevent incoming connections except to certain well-known machines and ports. If you can configure these to include your VNC server, then you will be able to access it from anywhere in the world. There is a good argument to be made for the fact that VNC is less of a security risk than X, so if your site doesn't allow X in or out it may still allow VNC.

Many modern firewalls will allow outgoing connections initiated from inside, so you can often access servers on outside machines. It is straightforward, for example, to recompile the viewer source to include SOCKS support, or to make other special arrangements. See the contribs page.

It's a pity that Java within a browser doesn't automatically use SOCKS if the browser is configured to use it. There's probably Java SOCKS support out there somewhere...

If your internet access is through a router which does Network Address Translation, you may be able to configure the router to redirect particular incoming ports to particular machines. So you could run WinVNC with a display number of 0 on machine snoopy, and with display 1 on machine woodstock, then set your router to send port 5900 to snoopy and 5901 to woodstock. See below for information on the other port numbers used by VNC.

Q. Which TCP/IP ports does VNC use?

General Questions:

Q. How do I make VNC go faster?

We find VNC to be perfectly acceptable as our normal method of accessing Unix desktops on a daily basis. This is over a 10 M/bit ethernet on reasonably modern machines, using the X or Win32 viewer. Because Windows gives us fewer hints about what it's doing, and because we don't have the source code for Windows, the NT server has to work harder to find out what's changed, and so a really fast machine should make a big speed difference. For more information about how the Windows server works, see the WinVNC documentation. But if you've been disappointed by the speed of the Windows server, don't give up. We're improving it gradually, though it'll be a while before it's as fast as on Unix.
There are several things that can slow any VNC session down, however, and you may like to consider these if you find it too slow:

- Unusually 'busy' desktops. The VNC protocol is very efficient at rendering areas of a single colour, such as you generally find on window title bars, scrollbars, backgrounds of pages etc. But if, for example, you have pretty 24-bit photographs of your girlfriend as your screen background, or dithered title-bars on your windows, you may pay a price for the aesthetics. **A colourful or patterned desktop background will probably slow down VNC more than any other single factor.** We have some suggestions on speeding up the twm window manager, some of which will also apply to other environments.
- Hi-colour desktops. Don't use 24-bit colour if you can use 16 or 8 equally well. Remember, on Unix you can run multiple servers, so I have a big 16-bit desktop for normal work and a small 8-bit one for when I log in from home. The server can send out a wide range of pixel formats, and some viewers will allow you to request a specific format for that session. On the Windows viewer, for example, if you click Options... when making the connection, you can request only 8-bit pixels from the server - useful if the network gets slow. If you are using a modem, I recommend changing the shortcut in the Start menu to include the /8bit option - this will then be the default. Similarly, if you regularly connect to a remote WinVNC server, consider whether you could run happily at lower resolution. A 1280x1024 screen has more then 4 times as many pixels as a 640x480 one, and if all you are doing is checking a printer queue you probably don't need them all! Note, though, that on WinVNC, 16-bit colour is usually the best to use. See below.
- Elderly graphics cards or drivers may make quite a difference; this is a graphics-intensive application! On Windows the graphics system on the server will affect the speed as well as the one on the viewer.
- Some applications are not very economical about redrawing their display. Early versions of Unix Netscape, for example, tended to draw everything twice when scrolling, which did nothing to help the smoothness under VNC. X11Amp flashes its display very fast when in 'pause' mode.
- Some Java Virtual Machines are particularly fast at reading from the network and particularly slow at drawing to the screen, or vice versa. With the Java viewer it is worth experimenting with the encodings

[redacted]he contribs page for details.

Q. Are you going to make it more secure?

While we do hope eventually to add better security to VNC, there's also a good argument for not doing so. If security is a concern, it can be better to use a single system such as SSH or FreeS/WAN to encrypt all your traffic, rather than relying on the individual packages to do the right thing. Then, if you decide that one system is too easily crackable in a year's time, you can replace it yourself and all of your communications will benefit.

Q. Are you planning support for AIX, EPOC,HP-UX, SGI, Win 3.1, or my favourite platform ?

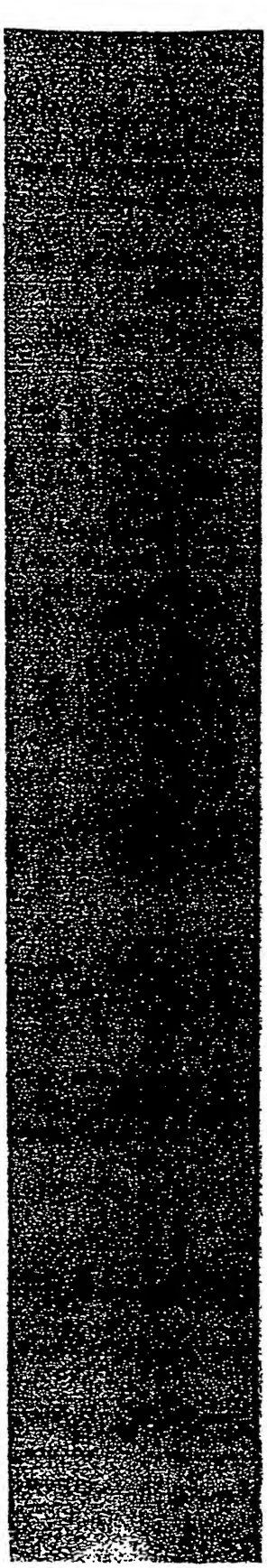
We have provided VNC on all the platforms we use here, and it's difficult to provide binaries for anything we don't have, and it takes a while to get up to speed on new platforms. Remember that a viewer is available for any platform which runs Java, though the speed may vary quite a bit. But for many platforms it should not be difficult to compile at least the viewer. If anyone tailors the sources for a particular platform we will happily either incorporate the changes in the main source distributions or make the patches available from our site. Information about other people's ports of VNC to a large number of other platforms can be found on the 'contribs' page.

Q. Would things work better if you compressed the stream?

VNC incorporates really quite efficient compression in the sense that we generally send a tiny fraction of the raw data, probably something like 1/20 on average. The details are in the protocol spec if anyone's interested. On a couple of test screen dumps we found that the Hextile encoding was more efficient than GIF! I don't know whether this is true in general.
But we haven't done more general encoding after that; we've tended to the view that (a) it might introduce too much latency and (b) most modems compress pretty well anyway. We are planning some zlib-compression experiments in the near future to see how this affects things. Because different bits of the screen can be sent using different encodings, the server could, in theory, detect that one bit would be most efficiently sent as JPEG, while another would be better hextiled.
The question is always how much work it's worth doing at the server to find this out. To some degree you can control this already, because the viewers allow you to specify your preferred encoding. Under X, if your viewer and server are on the same machine the viewer will use the raw encoding by default, otherwise it will use hextile. You may find that by selecting different encodings on the command line you get better performance.
See the suggestions above about using ssh, which also provides compression.

Q. Have you thought about caching bits of the screen at the viewer end?

Yes, that could also be good. You could have an off-screen cache in the viewer and the server could copy things from there to the screen. Management of this would add a certain amount of complexity, though. Since there is already a CopyRect primitive in VNC, an alternative approach would be to copy updates from another part of the screen if they already exist there, rather than resending them. Again, to make the server find out efficiently when this is worth doing would be an interesting challenge, and volunteers for



m[redacted]portant one is 59xx, where xx is the disp[redacted] number. The VNC protocol itself runs over this port. So for most PC servers, the port will be 5900, because they use display 0 by default.
In addition, VNC servers normally have a small and very restricted web server built in, which allows you to connect a browser to them and use the Java viewer. This runs on port 58xx. Note that this is the HTTP port used for downloading pages and applets, but once the applet is running it uses 59xx for VNC just like any other viewer.
The servers can be changed to listen on other ports if, for any reason, these are not suitable for you. See the server's documentation for more details. Most of the viewers, if given a display number larger than 99, will interpret it as a direct port number and will not add 5900.
If you are running a viewer in 'listening' mode, where it accepts connections initiated by the server, it will listen for incoming VNC on port 5500. It's unlikely that this will apply to you, because the servers we distribute publicly don't have this facility.

Q. How secure is VNC?

Access to your VNC desktop generally allows access to your whole environment, so security is obviously important. VNC uses a challenge-response password scheme to make the initial connection, but after that the data is unencrypted and could, in theory, be watched by other malicious users, though it's a bit harder to snoop a VNC session than, say, a telnet, rlogin, or X session. Since VNC runs over a simple single TCP/IP socket, it is easy to add support for SSL or some other encryption scheme if this is important to you. Axel Boldt <axel@uni-paderborn.de > suggests:

> SSH allows you to redirect remote TCP/IP ports so that all traffic is strongly encrypted, and this can be combined with VNC:
> Run sshd and vnc server on machine "server", then connect to it using `ssh -L 5901:server:5900` from client (assuming vnc server listens for connections on port 5900) and point your vnc client to client's own port 5901. Everything will be completely secure. [See the section in this FAQ about port numbers if you don't understand the 5900/5901 references.]
> Sshd and ssh are free for unix but payware for Windows. See http://www.cs.hut.fi/ssh/ for info about SSH.

Setting up SSH is pretty straightforward on Unix machines, but generally harder on Windows. Miroslav Luptak <Miroslav_Luptak@snt.sk> has posted detailed information to the mailing list about how he set his SSH system up on NT. See http://www.uk.research.att.com/vnc/archives/1999-03/0060.html for details. There is also a free SSH client for Windows available at http://www.doc.ic.ac.uk/~ci2/ssh .
SSH can also compress the encrypted data - this can be very useful if using VNC over slow links. Another minor issue you should know about if using the Unix viewer to connect via SSH: By default, when the viewer connects to a server on the local machine, it uses the 'raw' pixel encoding because this generally gives better performance for local access. If this 'server' is actually an SSHD redirecting the data to another machine, you probably want to override this using the -hextile option to the viewer, or you will send a lot more data over the network than you need to.
While we're on the subject of security, you should also be aware that only the first 8 characters of VNC passwords are significant. This is because the 'getpass' call used in the Unix server to read a password has this restriction, and the other platforms have been made compatible with this.
Ray Jones <rjones@pobox.com> has built a version of VNC which uses

Q. Can I use VNC over a modem without TCP/IP?

Not at present, but there's no reason why it shouldn't, and we realise that this would be useful. VNC could run over other transports such as RS232, firewire, USB, modems etc, but at present we rely on TCP/IP. This means that you can use VNC over anything which supports TCP/IP, so using it over a modem is just the same as any other network, once you have Dial-Up Networking set up. If you need to communicate directly between two machines without going via the internet/intranet, then set up a remote access server on one and dial in from the other.

We don't, at present, support simple dialup without TCP, chiefly because it involves writing a lot of code to talk to the modem. But it's on the 'suggested projects' page, so any volunteers welcome...

Q. Does VNC have any Y2K (Year 2000) bugs?

The WinVNC server and Windows viewer have been tested on a PC with its date running through the 2000 boundary without any problems, so unless the underlying OS or BIOS has difficulties, VNC on a PC should be fine. The VNC part of the X-based Unix VNC server only uses dates when writing the log files; the logfile entries are timestamped with a two-digit year, but the format is easy to change if required and the entries are not intended to be machine-readable. The developers of the XFree86 server on which Xvnc is based state that there are no Y2K problems (see http://www.xfree86.org/FAQ/). We therefore issue the standard disclaimer: we believe the VNC code, in its entirety, to be free from Year 2000 problems, subject to the other components of the systems on which it is running.

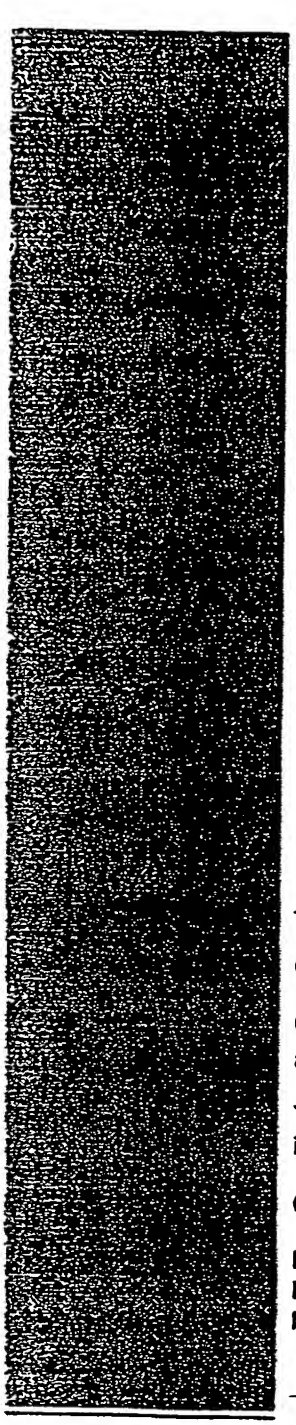
Q. How can I install WinVNC on multiple machines?

When you run WinVNC for the first time on a machine, it will prompt you for a password. If you are doing this on a large number of machines, especially remotely, this can be a nuisance. The way to bypass this is to make sure that the target machine already knows the password by putting the encrypted version straight into the registry before installing. So:

- Install WinVNC on a master machine and set the password
- Copy the appropriate registry entries from `HKEY_CURRENT_USER\Software\ORL\WinVNC3` or `HKEY_USERS\.DEFAULT\Software\ORL\WinVNC3` (if you installed WinVNC as a service) and install them on the target machine using your favourite registry utility.
- Copy the VNC files (typically under C:\Program Files\ORL\VNC) onto the remote machine.
- Install as a service or into the startup directory as appropriate.

Q. Can I connect multiple users to the same Windows server, and have them each see their own desktop, as with WinFrame, NTrigue, WTS, etc?

No. Windows NT has a reasonable concept of multi-user access, but not where the GUI is concerned. Basically, you can't do this without access to the source code of Windows, and that's rather difficult to get hold of! We have successfully run multiple WinVNC servers on a Windows Terminal Server machine, but they don't update unless a WTS client is also connected, which rather defeats the purpose. It may be possible in future to get this working



[redacted]urse, if your server is Unix-based, then y[redacted]ave no problem. You can run dozens of VNC servers on a single machine.

Q. Any other tips?

Several people have indicated that they have to use Windows occasionally but prefer to use Unix most of the time, and so want to access a PC under the desk from the Unix box.
Here's a suggestion: all other things being equal, I recommend using the Windows box to view the Unix machine rather than the other way around. This is chiefly because Windows generally works better as a client than as a server, and also because PC graphics cards are often better than those in Unix workstations. Remember, you can create a VNC session of any pixel depth you like. For my day-to-day work I use a Solaris 16-bit VNC session displayed on my PC. Few of our Sun machines have more than 8-bit color hardware, but the PC has plenty of bits to spare.
If you're very anti-Windows you can make your VNC desktop the same size as the screen and set the taskbar to 'Auto hide' and just pretend you're on an X terminal, but pop up the Start menu when you have to use PowerPoint.... The Windows viewer 3.3.2R4 and later has a proper 'full-screen mode', so you don't even need to bother with auto-hide.

Q. You misspelled 'organization' on the download page!

No we didn't. We spell it like that in the UK. Actually, we spell it both ways, but the 's' spelling is more common, despite what the OED says! Now, as for 'misspelt'...

Compiling the source

Q. I'm trying to compile WinVNC and the compiler complains about various missing files!

You need to compile using the No_CORBA configuration, or it will try to include various files which are part of our internal version only.

Q. I'm having trouble compiling VNC on my platform...

Have you checked the contribs page? Several people have provided hints on how to build VNC on other platforms. If yours is not listed there, you might at least get some clues..

Virtual Network Computing

AT&T

Downloading VNC

The VNC system is available for general use under the conditions of the GNU General Public Licence . You should be aware of the terms and conditions of this licence, which is also contained in the distribution itself.

Please fill in the form below to download VNC. It's not required, but we'd be grateful if you filled it in. We're just curious about who's using it and what they're using it for. This information is purely for our own VNC-related use and will not be disclosed to any other party. Neither will we, in general, send any mail to any address given here. If you want to keep in touch with VNC developments after downloading, please join the mailing list.

This page is part of the downloadable documentation; if you are looking at a local copy, please check the latest online version - things may have changed!

There is some other great free software which is downloadable from our site.

VNC download

Your Name:

Your email address:

Your organisation:

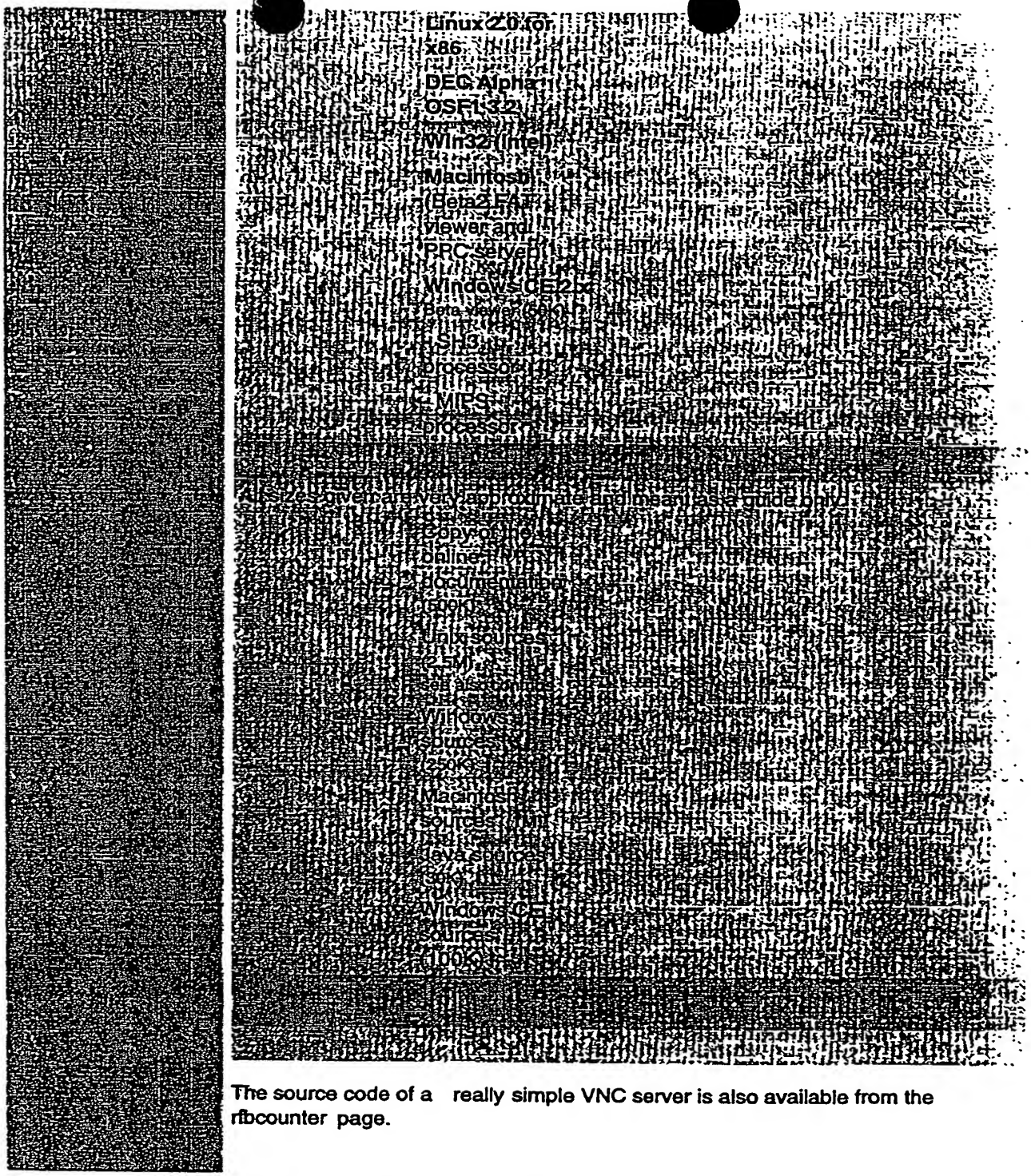
Comments:

We would be interested to hear any comments about your interest in VNC, your expected use, how you heard about it, etc.

These include the server and viewer for a particular platform. They all include the Java viewer. Each binary package is very approximately 1 Mbyte in size and the tgz version is usually the smallest. See the release history page for details of recent changes.

If your platform isn't listed here, somebody else may have ported VNC to it. Have a look at the contribs page.

Please choose the package(s) of interest, you can always come back for more later!

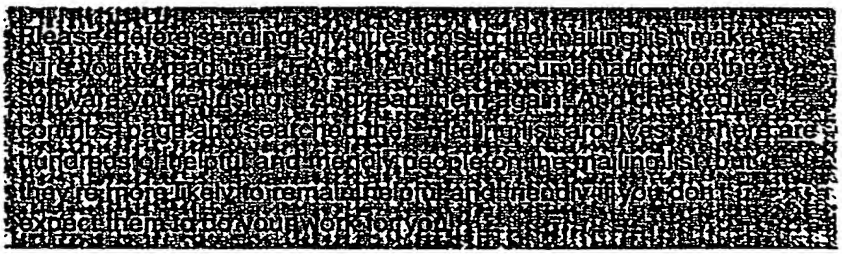


The source code of a really simple VNC server is also available from the rfbcounter page.

Virtual Network Computing

AT&T

Keep in touch with VNC developments



There are several ways to keep in touch with developments in VNC:

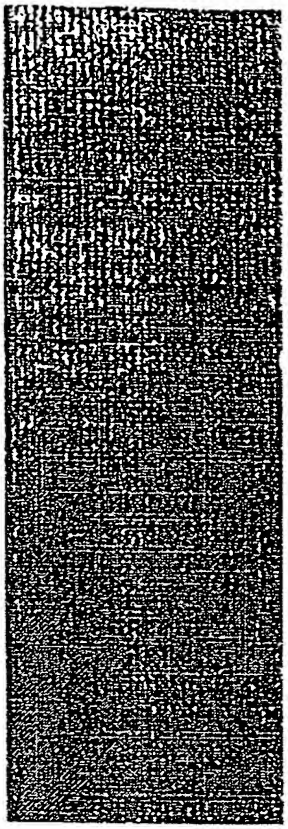
- Join a mailing list. This is the most important. If you don't know what a mailing list is or how it works, have a look at this introduction. There are 3 different mailing lists - you can choose the one(s) most approriate for your level of interest!

vnc-list	Main list for questions, suggestions, bug reports etc.
vnc-list-digest	This carries the same messages, but will just send you one message a day containing all of the last day's messages. You may prefer this to the standard list, particularly if your mail reader won't filter the traffic into a separate folder for you.
vnc-announce (new)	This is for announcements of new versions of VNC, successful ports to other platforms, availability of related software. This list is 'moderated', which means that messages sent here will only appear if we approve them, which we will do provided they are appropriate for the list.

You can add yourself to or remove yourself from the mailing lists by sending commands in the **body** of an email to majordomo@uk.research.att.com. For example:

```
subscribe vnc-list
```

This asks for your name to be added to the list. Replace 'vnc-list' with the list of your choice. More details will then be sent to you by



'unsubscribe' instead of 'subscribe'. [redacted] can also just send the command '`help`', to get more detailed instructions. Note that these should be sent to 'majordomo' and not to the list itself! If your mailer modifies your email address on the way out, you will be subscribed under this modified name, which can make unsubscribing difficult, particularly if the mailer changes it to something different at a later time, so if you like you can specify the email address to be used:

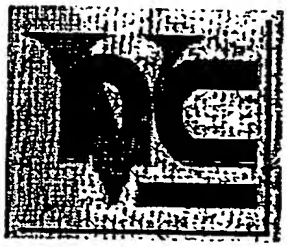
```
subscribe vnc-list myaddress@company.com
```

and the same applies to unsubscribing.

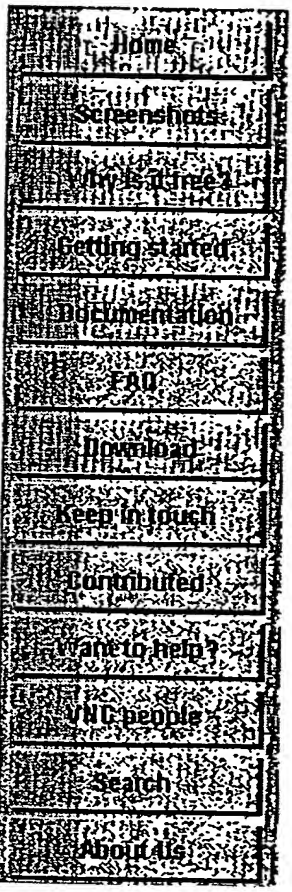
- The archives of the mailing lists are available here, and you can search them here.
- Check these web pages regularly. If you're looking at a mirror of the main VNC site, click here for the master version.
- Darxus@op.net has created a VNC channel in IRC. For information on how to join it, see his web page.
- If you need to contact the developers directly, send email to vnc@uk.research.att.com . NOTE: Because of the level of interest in VNC we cannot, in general, reply to messages, though we do promise to read them all! The mailing list is a much better method of contact; others may have solved your problem before, and may be able to help if we're not available.

We're interested in any and all feedback, but new features & bug fixes are much more likely to be incorporated if you send us sample code showing how to accomplish them!

For comments, feedback, etc, please see the 'Keeping in touch' page.



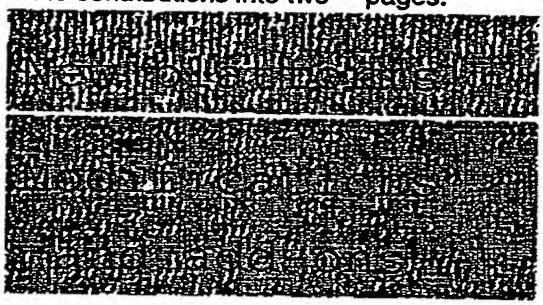
Virtual Network Computing



VNC Contributions

Many people have contributed to the VNC project, with ports to new platforms, and various other modifications and extras.

We have divided the contributions into two pages:



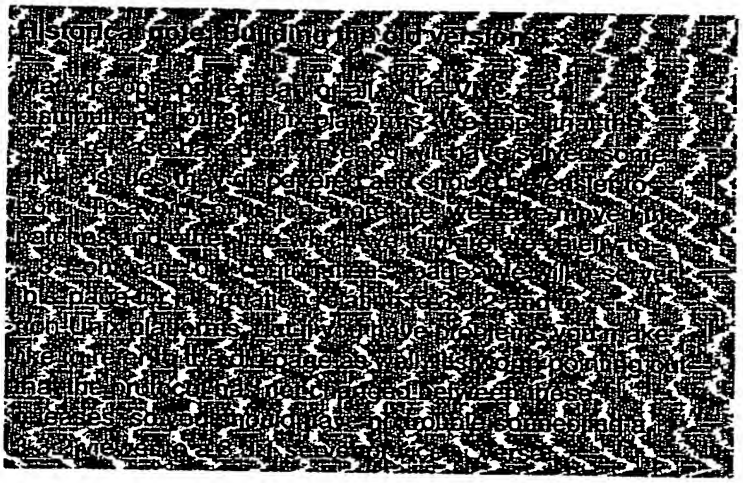
For comments, feedback, etc, please see the 'Keeping in touch' page.

Virtual Network Computing

AT&T

VNC on other platforms & environments

We prefer not to distribute binaries for platforms we cannot test here, but several people have contributed source code modifications or hints to allow VNC to compile under different platforms. Please note that we cannot officially endorse or be responsible for these but we are grateful to all the contributors. We may try and incorporate some of these in a future source code release. If you port VNC to a new environment we would be grateful to hear about it; please post such announcements to the mailing list.



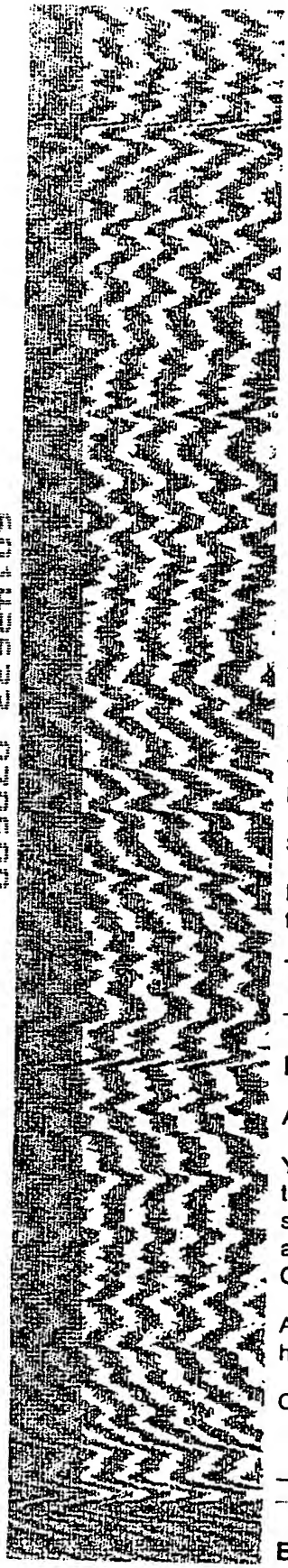
Linux RPMs & Debian packages

Other people have kindly packaged VNC up in RPM and Debian package form; see for example under 'V' at the RPM archive http://rufus.w3.org/linux/RPM/ and at www.debian.org . Note that these may not be identical to the standard AT&T distributions, and may not be updated as frequently, so you should check the main VNC site frequently.

AIX

Chuck Hines <chuck_hines@VNET.IBM.COM > writes:

For anyone trying to build the latest version of VNC under AIX 4.1.5 may be interested in the patch below which contains the quick and dirty changes I needed to do to be able to comple and link successfully (hopefully I didn't miss anything). Things went pretty smoothly after these minor changes and it seems to be running fine.Basically the patch adds the necessary AIX sys/select.h inclusion where needed, removes the X11R6 specific stuff, and forces



sys/resource.h to be included. That last one left me sort of puzzled,as the way it was trying to build should have not tried including it at all (which should have been fine), but it looked like os/osinit.c WASN'T including it while os/utils.c WAS for some unknown reason (andthe #if logic looked correct to me in both files) creating undefinedsymbols.

– Chuck

Chuck's original patch is available here: vnc-3.3.2-aix.patch.txt and he has now set up a small web site with more details and binaries available. See http://www.idsi.net/~bshma/chuck/vnc.html .

Acorn RISC OS

We know of two viewers for RISC OS.

The first, created by simon@bigblue.demon.co.uk, is available from http://www.bigblue.demon.co.uk/VNC.html .

The second, by Leo White <leo@brighteyes.u-net.com > is at http://www.brighteyes.u-net.com/ .

A ***server*** for RISC OS is also available from http://www.interconnex.co.uk/~paul/

Amiga

VVA - a VNC viewer for the Amiga. which was started by J örg Dietrich is now being maintained by dspach@cybercable.fr . See his web page for more details.

Stéphane Guillard - stephane.guillard@steria.fr writes:

I have set up a (incomplete, but usable) Amiga VNC server, which can be found on Aminet (http://de.aminet.net), as comm/tcp/AmiVNC.lha (ie. here).

This is a work in progress.

BeOS

Andreas F. Bobak writes:

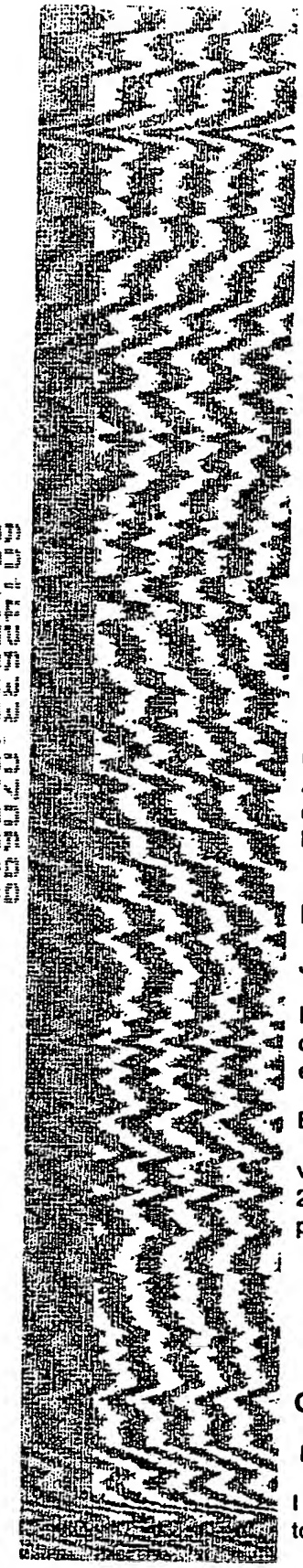
Yesterday, I made a first port of the VNC viewer to the BeOS. I mixed together the Win32 viewer and the Unix viewer and added a bunch of BeOS specific stuff. It basically works but perfomance does scream. RRE encoding is broken and Hextile encoding has a strange bug, but it's usable with just CoRRE and CopyRect.

A snapshot of the sources and a x86 binary can be found under http://abstrakt.ch/be/

Cheers
-boby

– Andreas F. Bobak bobak@relog.ch

BSDI



Kurt Seel <kseel@utcorp.com> writes:

vnc compiles cleanly on bsdi 3.0 (no patches) with the following pecuiarity - socket.c and httpd.c had to be ifdef'ed like so :

```
#ifdef __bsdi__
#undef _ANSI_SOURCE
#endif
#include <sys/time.h>
#ifdef __bsdi__
#define _ANSI_SOURCE 1
#endif
```

It seems to work fine. The switch to xfree 3.3.2 really did some good here!

Cygwin32

Valery Tulnikov has built the server and viewer under Cygwin-32, based on the 3.3.1 patches by Sergey Okhapkin. This allows you to run the X viewer and server under Win32. Yes, there are some good reasons why you might want to do this! See http://www.dol.ru/users/valtul/ for more info.

DOS

Marinos J. Yannikos <mjy@pobox.com> has written a VNC viewer which runs under DOS, using packet drivers and the Waterloo TCP/IP library along with the Allegro graphics library. The whole system including the IP stack fits comfortably on a floppy disk. You can get it from http://www.complang.tuwien.ac.at/nino/dosvnc.html .

FreeBSD

Joe Evans <evans@ittc.ukans.edu> reports:

I compiled the VNC stuff on FreeBSD 2.2.5, and it seems to work fine. The only compile glitch was that you need to remove the gnumalloc library from the extra libraries list in order to do the link step.

Bruce Mah < bmah@ca.sandia.gov> adds:

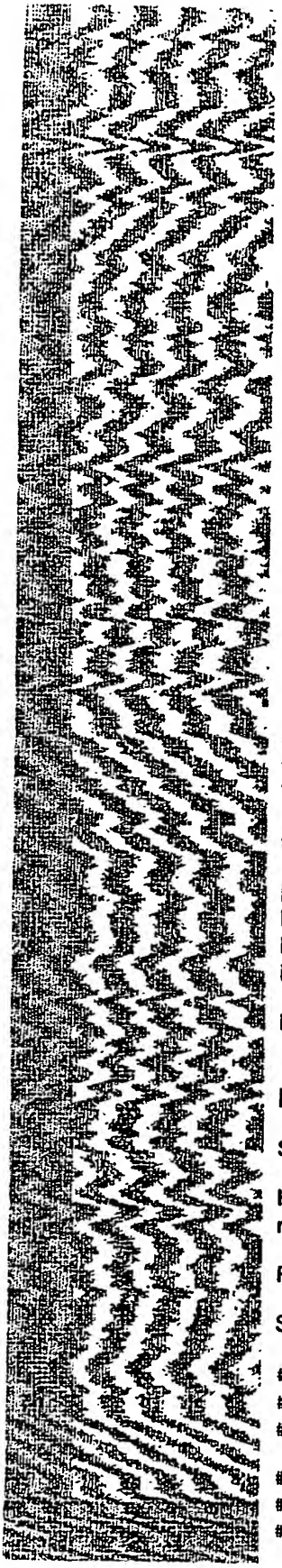
vnc is now a part of the FreeBSD ports collection...on FreeBSD 2.2.7-RELEASE or newer with an installed ports collection, the installation process is simply:

```
cd /usr/ports/net/vnc
make install
```

Geos (eg. Nokia 9000)

Marcus Groeber < mgroeber@compuserve.com> writes:

I have just uploaded the first public version of the VNC Viewer for Geos to my home page. Have a look at



http://ourworld.compuserve.com/homepages/mgroeber/nokia.htm

It allows you to access VNC servers from machines running Geos 3.0, which would be either a Nokia 9000/9000i/9110 Communicator, a desktop PC running NewDeal Office 3.0, or a Brother GeoBook notebook.

Enjoy!

GGI

Steve Cheng steve@ggi-project.org writes:

libGGI is a portable graphics library with a flexible design. See <http://www.ggi-project.org > for details. The graphics application can betransparently "retargeted" to different types of displays including X11, Linux svgalib and fbcon. (Win32 soon to come.) The VNC targets adds the VNC protocol to this list. You can even run doom over VNC! :-)

Standand VNC clients can be used with the VNC target/GGI application as the server.

You can get it here: <http://shell.ipoline.com/~elmert/vnc.tar.gz >

Untar it under degas/lib/libggi of the GGI devel tree. (The stable GGI tree won't work because of namespace changes,etc.)

A long while ago someone wanted a "stripped-down version of the server part - one that skips listening/authentication phase and just uses stdin/stdout for the communication (run-once application)." This is not currently done yet, but I have made it easy to do so. The only problem is the libGGI application trying to use stdin/stdout.

[The VNC team] have asked me not to distribute this as part of libGGI (yet). (It has GPL'd code; the other parts are LGPL as the rest of LibGGI). So it won't be
in the GGI CVS tree now.

Bug reports, fixes, and feedback welcome.

HPUX

Several contributors here. Karl Hakimian <hakimian@aha.com> writes:

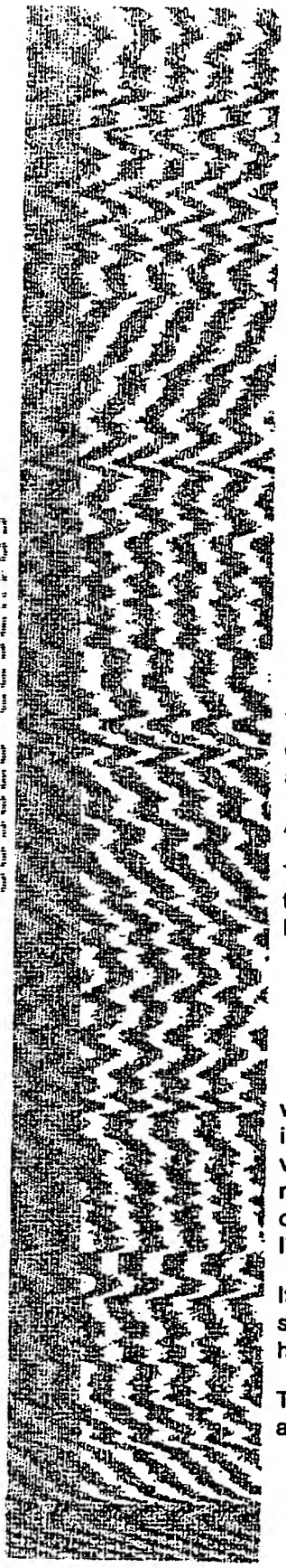
HPUX did not go as smoothly as some of the other OS's that have been reported, but I did manage to get things to compile under hpux 10.20.

First I had to change Xvnc/config/cf/hp.cf

Same as for 3.3.1, I made sure the following were set

```
#define ExtensionOSDefines -DHPINPUT # -DXTESTEXT1
#define XhpServer NO
#define BuildXInputExt NO

#define BuildPex NO
#define BuildPexExt NO
#define XvncServer YES
```



I also had to change the following to NO
`#define NeedBerklib NO`

That got things most of the way compiled. I then could not link Xvnc becuase of several missing objects

```
limitNoFile
limitDataSpace
limitStackSpace
```

Turns out I just needed to include sys/resource.h to two files in the programs/Xserver/os directory, the following patch takes care of that. (Karl's short patch is here)

Karl Hakimian
hakimian@aha.com

And Mike Cooke writes:

Just to inform you I've managed to build the vnc suite on the following HP box
HP-UX <name> B.10.20 E 9000/879

I had major problems trying to build it with the standard HP tools and after much head banging and source editing, I decided to forget it and switch to gcc which worked after about the 3'rd build. I applied the patches as advised in the contrib section on your site.

The only further problem I had was getting the Xvnc server to recognize the correct fonts - somehow the aliasing config here seems a bit odd, so to get around that problem I just ran the x font server and it solved all the problems.

And Ulrik Dickow <ukd@kampsax.dk> adds:

This evening I succesfully built vnc-3.3.2r3 on HP/UX 10.20 9000/712 with the
HP ANSI C compiler. I first naïvely tried

```
xmkmf
make World
cd Xvnc
make World
```

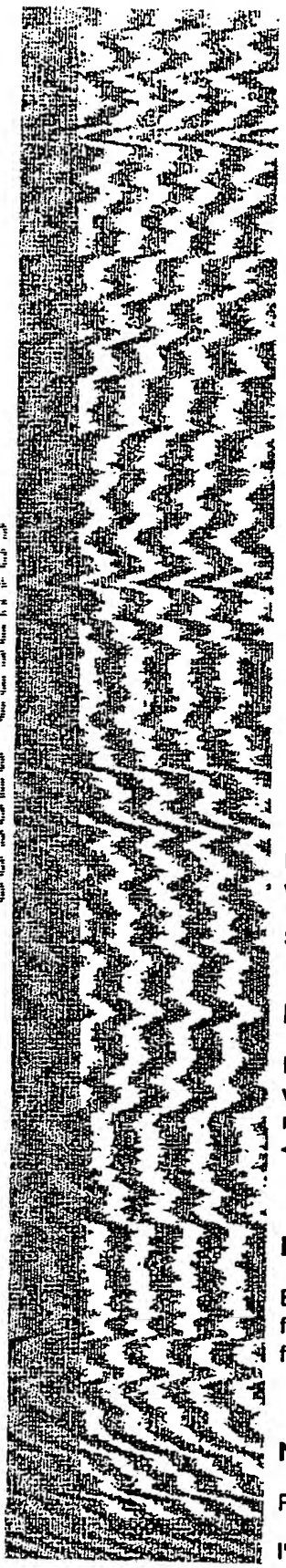
without any modifications to the source files, but this gave lots of errors in the Xvnc compilation. Then I changed hp.cf with Karls modifications from vnc/platforms.html and changed the two C files that his "short patch" mentions (but in a slightly improved way). This compiled succesfully, with only harmless warnings. To make it easier for others to build on HP/UX 10, I've attached the resulting "jumbo patch" to this mail.

It would be even easier if the changes were permanently done to the master source. They don't affect other platforms than HP/UX, and I guess they'll help HP/UX 9 and 11 users too, won't they?

To complete the help for fellow HP/UX'ers, the second attached patch modifies a few lines of the vncserver Perl script to

a) start a CDE session if possible, otherwise fall back to the old code

b) set the `-fp' argument reasonably for HP/UX (the CDE script will add more to this path).



Due to b), the second patch should "not" be applied to the general distribution, although a) might be.

If somebody have problems with these plaintext unified diffs, I can provide context diffs instead, possibly in base64 or uuencoded form. I start out with unified diffs, since they are much shorter and much more readable in this case. Apply them in the top vnc source directory like this:

```
patch -p1 < vnc-3.3.2r3_unix.ukd-kh-hpux-patch
patch -p1 < vnc-3.3.2r3_unix.ukd-hpux_specific-patch.txt
```

Long live GNU patch(1) (http://www.gnu.org/).

Thanks to ORL for conceiving a great product, and to Karl for finding out how to compile it on HP/UX 10. BTW, I compiled on a machine with these bundles installed, as part of an upgrade from 9.0x:

```
B3393AA_APZ B.10.20.02 HP-UX Developer's Toolkit for 10.0 Series 700
B3898AA_APZ B.10.20.02 HP C/ANSI C Developer's Bundle for HP-UX 10.20 (S700)
B3910BA_APZ A.01.00 HP aC++ Compiler S700
```

Ulrik's patches are at
http://www.uk.research.att.com/vnc/contrib/vnc-3.3.2r3_unix.ukd-kh-hpux-patch
and
http://www.uk.research.att.com/vnc/contrib/vnc-3.3.2r3_unix.ukd-hpux_specific-patch.t

KDE

Markus Wuebben markus.wuebben@kde.org has built a version of the viewer which fits nicely into the X-based KDE environment.

See http://studserver.uni-dortmund.de/~su0197/kde/kvnc/ for details.

MacOS (alternative)

Dair Grant <dair@webthing.net> has written an alternative Macintosh viewer which supports the Appearance Manager and Navigation Services, which means it looks more attractive on recent versions of MacOS. You can find it at <http://www.webthing.net/vnc.html >.

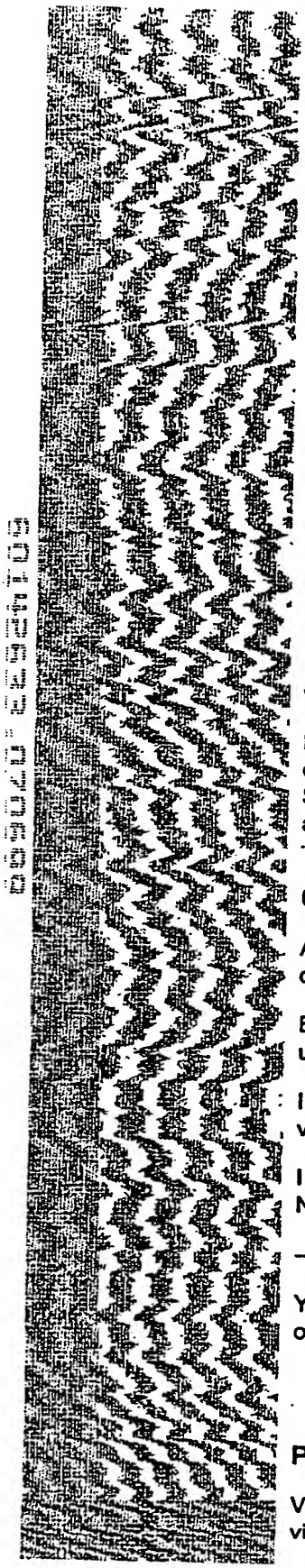
NetBSD

Era eriksson <era@iki.fi> wrote to tell us that information about a NetBSD port for i386 is available at
ftp://ftp.netbsd.org/pub/NetBSD/packages/pkgsrc/net/vnc/README.html

NetWinder

Ralph Siemsen <ralphs@netwinder.org> writes:

I've "ported" the server and client portions of your VNC package to our



NetWinder platform (an ARM-based linux system). There are only minor changes necessary that stem from the fact that your package tries to rebuild "imake" itself - but is isn't aware of the arm platform. I've attached the patch below; it is against vnc-3.3.2r3 for unix.
... We'll be providing binaries from our web site for NetWinder owners..

Ralph's patch is available at
http://www.uk.research.att.com/vnc/contrib/netwinder-patch.txt

Nokia 9000

See 'Geos' above.

OpenStep/Mach

David Young (dwy@picasso.eng.ace.net) writes:

I've written a client for OPENSTEP/Mach (that spiffy NeXT OS) for VNC.

It currently supports display at 24, 12, and 8 bpp, mouse and mostly-functional keyboard input (ASCII and control characters work; the mapping isn't yet complete and I'm looking for suggestions on how to complete it), and some other client-side niceties. Encodings other than raw are on their way, as is NSPasteboard integration.

I'm looking for users (preferably with VNC and OPENSTEP experience) who can bang on this client on original NeXT boxes, Intel machines, or SPARCstations running OPENSTEP 4.2, or just people who might find it useful at this early state.

OS/2

Akira Hatakeyama <akira@sra.co.jp>is working on a native PM viewer. You can get it from http://www.sra.co.jp/people/akira/os2/vnc-pm/index.html

Bosse Nystr öm bosse@postman.riken.go.jp has built the X viewer for OS/2 using XFree86. He writes:

I compiled the unix [3.3.1] sources with the attached diffs and got a working viewer under OS2 with XFree86 (and EMX).

I tested it with servers for OSF and Win32 (rev 16), some problems with National characters for the win server otherwise it works fine.

-- Bosse

You can get Bosse's version from his FTP server at ftp://bfs.riken.go.jp/pub/vnc/ , or from Ted Sikoras site at http://tsikora.tiac.net under XFreeOS2.

PalmPilot

Vladimir Minenko minenko@icsi.berkeley.edu has created a port of the VNC viewer for PalmOS 2.0 or higher.

You can get PalmVNC from http://www.icsi.berkeley.edu/~minenko/PalmVNC

SCO OpenServer

Ben Maizels <bmaizels@analystic.com> sent the patches he used to compile under SCO OpenServer 5.

You can find his message at
http://www.uk.research.att.com/vnc/contrib/sco-openserver.txt .

SGI Irix 6.2

Wolfram Gloger <wmglo@dent.med.uni-muenchen.de> writes:

The following small patch was all that was necessary for me to successfully compile Xvnc on SGI Irix 6.2 with the N32 binary format. A binary is available at ftp://ftp.dent.med.uni-muenchen.de/pub/wmglo/ .

Wolfram's patch is available at
http://www.uk.research.att.com/vnc/contrib/vnc-irix6.2-patch.txt .

SPARC Linux

James Hall <jhall1@isd.net> writes:

Just thought I would let you know: VNC 3.3.2r2 compiles just fine on Linux/SPARC. I am using it now to run Word and Lotus Notes without having to go back to my office and use my Windows 95 PC.

To compile, I just ran the 'configure' script, then typed 'make'. I think I got a few warnings during the compile (mostly unused variables) but VNC runs fine. This was on Red Hat Linux/SPARC 4.x, so I didn't have glibc. Don't know if that makes a difference.

SunOS 4.1.3

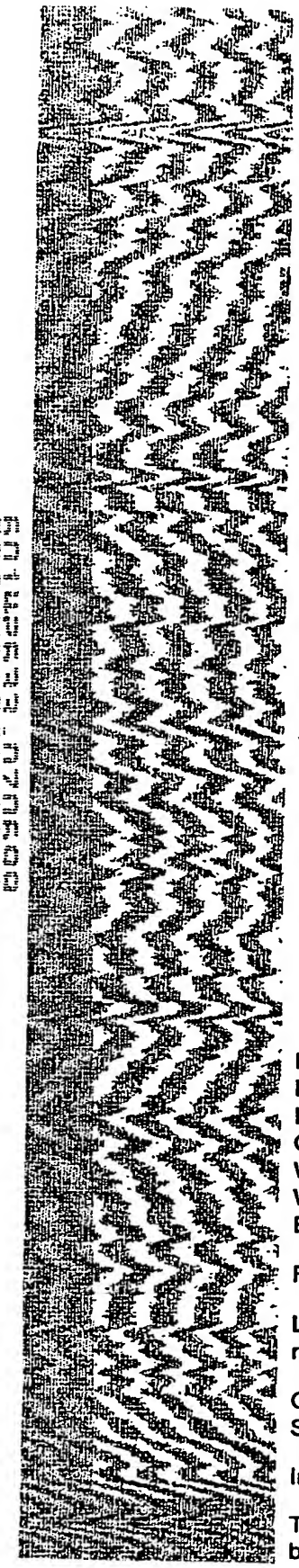
We've had reports that this buiilds without any problems if you use gcc.

SVGALIB (Linux without an X server) & Single-floppy Linux

Ganesh Varadarajan and Sitaram Iyer have built a vncviewer which runs from a Linux console using the svgalib library. You will need to install svgalib and configure /etc/vga/libvga.config for your graphics card. Try creating servers of different geometries and connecting to them - if your svgalib setup doesn't work for one resolution it may work for another. Persevere - this has a lot of potential, I think. Note that the current version will not generally be able to connect to Windows servers, because it requests a palette-based display which the Windows server cannot generate.

The authors wrote:

This is a alpha port of vncviewer to svgalib based on the [3.3.1] X client. You no longer need to start X to use vnc ! Even that old 4MB 386 which you've got in the corner can be used as an X terminal. It can also be used on a single-floppy Linux - not quite the answer to the QNX challenge, but good



enough for me.

Some hints on getting svncviewer working for you :

1. First of all make sure your card is supported by svgalib at the higher resolutions like 640x480x256. Otherwise you can only use 320x200x256
2. Sometimes svgalib incorrectly detects the card or doesn't detect it at all and defaults to the VGA driver. In this case you might have to edit /etc/vga/libvga.config (this is Redhat's svgalib config, your distribution might have it elsewhere) and uncomment the line corresponding to your chipset. Cards I've had trouble autodetecting: Cirrus Logic 5446 - uncomment the chipset Cirrus line. Trident TVGA 8900 - incorrectly detected as Mach64. uncomment TVGA. Chips and Technologies - add the line chipset C&T
   (Note: It seems some distributions don't have svgalib compiled with C&T support. You might need to recompile svgalib).
3. If you get something like read error: broken pipe, it means the remote Xvnc has closed the connection, usually because the client's parameters like bpp are not acceptable. Check out the Xvnc logs.
4. There is a workaround in svga.c which disables use of acceleration. You might want to comment out NO_ACCEL and see if acceleration works.
5. If your mouse doesn't work, make sure that /dev/mouse is rw-able by you. This is because svgalib opens the mouse O_RDWR and we give up suid perms immediately after vga_init and before opening the mouse. Redhat's default is 660, you might want to make this 666.

TODO:

1. The viewer is not flexible about selection of graphics mode. It selects the closest possible to the server geometry/depth. It bails out if a resolution = server geometry is not available.
2. Only the std. keymap works, for other keyboards you might have to hack keys.h which maps Linux keycodes (different from X keycodes) to X keysyms.
3. Mouse middle button doesn't work (svgalib problem ?)
4. Keyboard LEDs

Please mail bug reports/patches etc. to ganesh@cse.iitb.ernet.in
In your bug report, please include the following information:
Linux distribution, kernel version, svgalib version.
Graphics card and whether other svgalib apps work at *high* resolutions.
What parameters Xvnc was started with.
What parameters were used for svncviewer.
Exact error messages, snippets from log files where applicable.

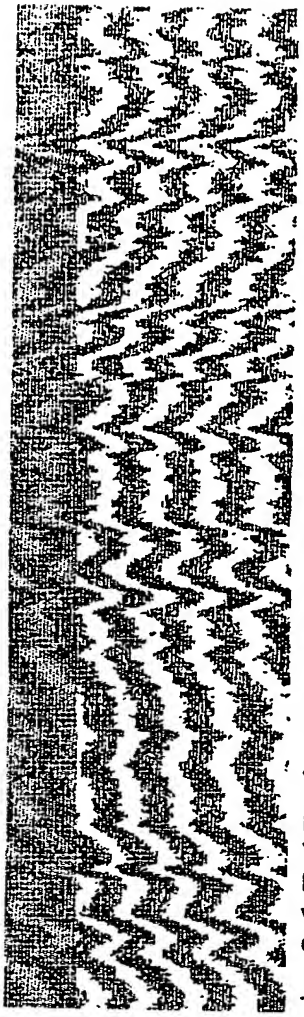
For the latest in svnc, check http://www.cse.iitb.ernet.in/~sitaram/vnc

Lastly, a BIG thank you to the ORL guys for making vnc freely available. VNC rules !

Ganesh Varadarajan <ganesh@cse.iitb.ernet.in>
Sitaram Iyer <sitaram@cse.iitb.ernet.in>

Indian Institute of Technology, Bombay.

The sources for svncviewer are here: svnc-0.1.tgz Remember that you will need both X and the VNC sources on your machine to build it, though you won't



need them to run it!

Single-floppy Linux

Karl Heinz Kremer khk@cyberdude.com has created a single-floppy linux distribution which includes svncviewer. You can now use an old 486 without even a hard disk as an X display. See http://www.stuttgart.netsurf.de/~khk/lods.html for details

VMS

VNC viewers for VMS 7.1 on both Vax an Alpha are available on the Law Bulletin Publishing Company's FTP server. Se the entry below for Windows NT/Alpha, and note the point about using a normal FTP client and not a browser.

Windows CE

The Windows CE viewer is now released! See the Download Page and the Documentation.

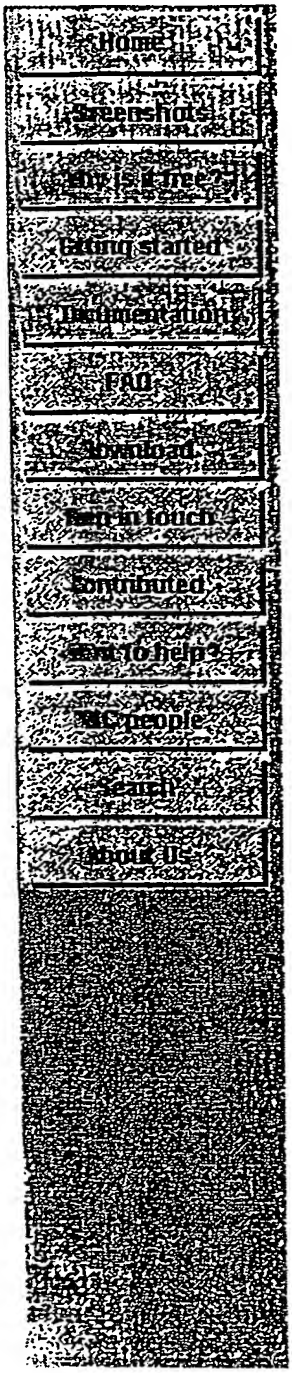
Windows NT/Alpha

John Ross Hunt <hunt@lbpc.com> writes:

Binaries and VC++ project files are now available for Alpha NT WinVNC3.3.2. You can download them from: ftp://ftp.lawbulletin.com/vnc/ You will probably have better luck downloading with a standard FTP client instead of using a web browser (it's a firewall issue). We plan to upgrade soon, but until then, the old-fashioned way works best.

—John Ross Hunt, Law Bulletin Publishing Company

Virtual Network Computing



VNC Extras, Add-ons and modifications

Firewalls

Karl Harkimian <hakimian@aha.com> sent in a patch to compile the Unix viewer so that it can reach external servers through a SOCKS firewall. It's very simple, and the patch is here: socks-patch.txt . These patches apply to 3.3.1 but they are simple enough to apply by hand to later versions.

Obviously, you cannot access a server inside a firewall from outside without specially configuring the firewall; that's what firewalls are for! Christian A. Lademann cal@zls.de has used an elaborate scheme:

I have had a problem with connections from my workstation to a PC connected to a customers LAN, because my firewall as well as the customers firewall does IP-masquerading. This means that my workstation can only connect to the customers firewall and the customers PC can only connect to my
firewall. The reason for this is, of course, to hide the complete LAN behind the firewall(s).

Christian's solution involved a script listening on a single accessible well-known port behind the firewall, which could accept incoming connections and redirect them to the appropriate machine. His full details are here: rvnc.txt

SSLeay encryption

Ray Jones rjones@pobox.com has built a version of VNC which uses SSLeay public key encryption for a more secure connection.

You can get it from http://web.mit.edu/thouis/vnc .

Restricting connections by IP address (TCP wrapper)

Wolfram Gloger <wmglo@dent.med.uni-muenchen.de> writes:

In the list archives i notice there was a discussion of implementing access control for Xvnc with the tcp wrapper library. I have now implemented this for 3.3.2r2. You will need to have tcpd.h and libwrap already installed.

Wolfram's full message is at

WinVNC. He writes:

... I wrote some code to read a list of IPs to allow and IPs to deny from a text file. The rules are identical (to the best of my knowledge) to /etc/hosts.allow and /etc/hosts.deny on my Linux box and the text file's syntax is close to that format. This is a feature which has been discussed at least once on the mailing list, so I thought that you might want to add it to the official code base.

The x86 binaries and source code are available at:

ftp://wik.res.cmu.edu/pub/vncip_bin.zip and
ftp://wik.res.cmu.edu/pub/vncip_src.zip

I have two new files, ipauth.h and ipauth.cpp. I made some changes to vncclient.cpp (but not the header) to use the class and disconnect unauthorized clients. I also have a sample "iplist.txt" file which contains the allowed IPs.

The format of the text file works like this:

```
<ALLOW|DENY> <Partial/full IP>
<DENY ALL>
```

An unlimited, unordered list of IPs (or partial IPs) may be entered into the file like this:

```
ALLOW 128.2.93.
ALLOW 128.2.87.80
DENY 128.220.
DENY ALL
```

In this case, the DENY 128.220. is redundant because of the DENY ALL, but you get the point. ALLOW ALL is the default, and if the user specifies that, it is ignored. Allows always take precidence over denies. This code is not case sensitive.

I am fairly sure that I got rid of all of my memory leaks (I ran Purify on it, but I have done some slight modifications since then). I also use the fstream library. I don't know if you consider this to be too much overhead.

-- Jared Smolens

zlib compression

Dave DeBarr (debarr@mitre.org) has modified the X server and viewer to use zlib-based compression. We plan to incorporate something similar in the standard release before long, but until then you can find his patches at:

http://www.uk.research.att.com/vnc/archives/1998-08/0039.html

In addition, Dave has provided patches for the Windows viewer at:

http://www.uk.research.att.com:80/vnc/archives/1998-08/0228.html

x2vnc

Here's a different twist to VNC. Fredrik Hubinette hubbe@hubbe.net has written a VNC-based variation on the popular x2x program. If you run x2vnc on an X server, you can move off the side of the screen and the mouse movements will then be sent to a VNC server (eg. a PC sitting beside it) He writes:

x2vnc is basically a stripped down version of the vncviewer but with slightly different goals and a very different GUI.. :)

x2vnc emulates a 'dual head' setup by catching when the user tries to move the pointer past the edge of the screen. This allows me to control both computers from one mouse/keyboard.

I have made x2vnc available for download from my web site:

http://www.hubbe.net/~hubbe/x2vnc.html

Enhanced Java viewer for JDK 1.1

Muddassar Farooq <mfarooq@cse.unsw.edu.au> has produced an enhanced version of the Java VNC viewer.

When run as a standalone application, it adds scrollbars, and the ability to specify 'host:display' as with the other VNC viewers.

It uses the Java 1.1 event model, so is 'better' Java, but won't run on older JVMs & browsers.

You can get it from
http://www.uk.research.att.com/vnc/contrib/mfviewer.tar.gz .

VncMonitor

John Wilson <tug@wilson.co.uk> writes:

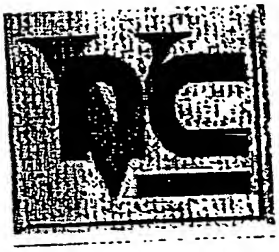
VncMonitor is intended for those people who need to monitor several remote systems. A single window is used to present all the displays. The tab or backtab key allows the user to switch between systems. The return key causes the currently viewed system display to be transferred to its own window and the user can interact with the system using the mouse and keyboard. Closing
the new window returns the monitored system display back to the initial window.

The configuration of VncMonitor is controlled by a file which contains all the information about what systems are to be monitored.

A version can be downloaded from:

http://www.wilson.co.uk/Software/vnc/VncMonitor.htm

that the dialup code could be external and easily customised.



Virtual Network Computing

AT&T

Home
Screenshots
What is VNC?
Getting started
Documentation
FAQ
Download
Keep in touch
Contributed
Want to help?
VNC people
Search
About us

VNC people

The following people, and no doubt others we've forgotten, are working on VNC or have contributed to the development at some point in its history, and we are grateful to them all. You can click on the pictures to go to their home pages.

Tristan Richardson

Quentin Stafford-Fraser

James Weatherall

Ken Wood

Andy Harter

Charlie McLachlan

Paul Webster

Printed copy of VNC website (www.uk.research.att.com/vnc) on Jun 20 1999.